Contract F61775-01-WE-030

Project title

Monitoring and information fusion for search and rescue operations in large-scale disasters

Final Report

"The Contractor, University of Rome,, hereby delcares that, to the best of its knowledge and belief, the technical data delivered herewith under Contract No. F61775-01-WE030 is complete, accurate, and complies with all requirements of the contract."

DATE: ______________________
Name and Title of Authorized Official: ___________________________________

(B) "I certify that there were no subject inventions to declare as defined in FAR 52.227-13, during the performance of this contract."
DATE: __________________________
Name and Title of Authorized Official: ______________________________________

August 2nd, 2002

Authors

Daniele Nardi, Fabrizio D'Agostino, Alessandro Farinelli, Giorgio Grisetti, Luca Iocchi

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE ***(DD-MM-YYYY)*** 20-08-2002	**2. REPORT TYPE** Final Report	**3. DATES COVERED** *(From – To)* 3 August 2001 - 03-Aug-02

4. TITLE AND SUBTITLE Information Fusion Research for RoboRescue	**5a. CONTRACT NUMBER** F61775-01-WE030
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Professor Daniele Nardi	**5d. PROJECT NUMBER**
	5d. TASK NUMBER
	5e. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Roma Via Salaria 113 Roma I-00198 Italy	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) EOARD PSC 802 BOX 14 FPO 09499-0014	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) SPC 01-4030

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report results from a contract tasking University of Roma as follows: The contractor will investigate artificial intelligence research methods for information fusion with application to search-and-rescue and large scale disaster relief. The objective is to develop and to deploy tools to support the monitoring activities in an intervention caused by a large-scale disaster. Particular focus will be on the software agents and the issues related to their development. The RoboCup-Rescue simulator will be used as the main development environment. Particular attention will be given to analyzing the information fusion problem, which concerns the way information acquired by different sources are gathered together and organized in a coherent manner.

15. SUBJECT TERMS

EOARD, Artificial Intelligence, Information Fusion, RoboRescue

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18, NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT UNCLAS	**b. ABSTRACT** UNCLAS	**c. THIS PAGE** UNCLAS	UL	86	Christopher Reuter, Ph. D.
					19b. TELEPHONE NUMBER *(Include area code)* +44 (0)20 7514 4474

1. Introduction

The whole objective of the project is to develop and to deploy tools to support the monitoring activities in an intervention caused by a large-scale disaster, with a particular focus on the software agents and the issues that their development involve. The RoboCup-Rescue simulator is used as a main development environment. Particular attention is given to analysing the information fusion problem, which concerns the way information acquired by different sources are gathered together and organised in a coherent manner.

In this document we report on the results of this one-year project, by addressing the 3 objectives set in the technical plan:

***Objective 1**: Setting the hardware and software systems necessary to use the RoboCup-Rescue simulator.*

***Objective 2**: Studying and developing both information fusion techniques, knowledge modelling techniques, and agent architectures.*

***Objective 3**: Evaluating the adaptability of the RoboCup-Rescue simulator to any intervention area.*

The development of the project has been carried out according to the plan. In particular, we can assess that Objective 1 has been fully achieved, while the basis for an effective solution to the information fusion task has been established in Objective 2 and the structure for the design and implementation of the prototype foreseen in Objective 3 has been designed and partially implemented. We have prepared a set of separate documents that describe in detail the results achieved in the project and that are integral part of this report. Below we present a summary of the results achieved for each objective, by providing referenced to the attached documents, we then discuss the prospects for future research and exploitation of the results achieved, and finally we provide a list of the recent publications of the project proponents that are related to the research field of the project and a reference list of the attached documents. An overview of the project has been presented in F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, Monitoring and information fusion for search and rescue operations in large-scale disasters, *Proc. of Information Fusion (IF 2002)*, Annapolis, July 2002 (see attached document 5)*.*

2. Results of the project

Objective 1: Setting the hardware and software systems necessary to use the RoboCup-Rescue simulator.

The RoboCup-Rescue simulator has been first installed on a local network of PCs. In order to improve the level of performance, a set-up has been made also on a high speed network of 4 interconnected computers (Myrinet), that has been installed in our Lab.

In Structure and working of the rescue simulator, A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, M. Salerno (see document 1 of the 6 months Interim Report), *the structure of the RoboCup Rescue simulator and details of the installation at DIS are described.*

Objective 2: Studying and developing both information fusion techniques, knowledge modelling techniques, and agent architectures.

The results related to this objective can be summarized as follows:

1) *Survey of the literature on information fusion, with a special focus on the use of agent-based approaches, that we consider well-suited for the rescue application domain.*
2) *Design and implementation of architecture and agent models for situation assessment in a rescue simulation domain.*

In Information fusion, A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, M. Salerno (see attached document 2 of the 6 months interim report)*, we first discuss various characterizations of the field of Information Fusion. We then address the application domains where information fusion approaches have been developed and the most important techniques and architectures deployed in such approaches. In particular, we distinguish three architecture levels where the fusion of information can take place: signal, feature and symbol level. While* *there are few approaches that can be classified as belonging to the symbolic level, we emphasize that the problem of "situation understanding", which is central in a rescue scenario, can be addressed only by addressing information fusion at all levels. We then describe, in more detail, the approaches to information fusion that deploy agent technology, which are of specific interest to the rescue domain. The document is concluded by a first attempt to model, at an abstract level, a rescue domain, highlighting similarities and differences with the approaches reported in the literature.*

In Architecture and agent modelling for situation assessment, F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, (see attached document 3)*, we address the problem of situation assessment in a scenario, which is dynamically and unpredictably changing, such as a rescue scenario. In this document we present the general approach, while its instantiation to the RoboCup Rescue domain is described in the attached document 4. The architecture of our multi agent system combines the requirements of deliberation and reactivity, that have been pursued by through the design of so-called cognitive agents, together with the ability to coordinate the behaviour of several agents that contribute to the assessment of the situation, while fulfilling their mission. More specifically, under the assumption that the agent can communicate according to some structure, possibly referring to different organizations, we model the basic features that characterize the behavior of an agent: the plans that allow the agent to accomplish its task, its information fusion policies, its capability to cooperate with other agents. The document is structured as follows. We first address the architecture of the agents, by characterizing their capabilities to acquire and integrate information coming from several sources, as well as their ability of accomplishing long term plans, and to react to the situation as perceived. We then present the basic feature of the Agent Development Kit (ADK), which provides a basis for the design of agents that are capable of executing plans, combine information coming from several sources and cooperate in order to accomplish a common goal. Subsequently, we present the main features of the implementation of our ADK, specifically addressing the plan specification and the information fusion component.*

Objective 3: Evaluating the adaptability of the RoboCup-Rescue simulator to any intervention area.

With respect to this objective, we have devised and implemented two extensions of the RoboCup simulator, the GIS editor and the ADK, that make it suitable for experimentation in a scenario built from real data concerning the earthquake in Marche and Umbria in 1997 (kindly provided by the VVF of the Italian Ministry of Internal Affairs). Using these tools, we have reconstructed a simulation scenario and we have developed a simple model of the rescue agents based on the ADK.

In Adaptability of the Rescue Simulator F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, (see attached document 4, which is a revised and extended version of document 3 of the 6 months interim report)*, we describe the results achieved in objective 3, by describing the tools implemented for facilitating the use of the RoboCup simulator and the models built through them.*
The first tool is a GIS editor, that enables the use of the simulator in domains that are taken from data concerning real world scenarios. The GIS editor allows one to build (or complete, depending on the availability and the detail of geographic data on the chosen scenario) a map of the rescue domain. The editor outputs the map in a format that is suitable for use in the rescue simulator.
The second tool is the specialization of the cognitive Agent Development Kit (ADK) to the RoboCup Rescue Simulator. The ADK enables the programming of a simulation scenario with a pre-defined agent structure, by making it easy to handle situation information at various representation levels, by providing built-in tools for information fusion and action planning, as well as to accomodate different communication structures and coordination protocols.
The GIS editor helped us in reconstructing a simulation scenario of the city of Foligno related to the earthquake of Marche and Umbria in 1997. In addition, we have applied the ADK to design some elementary agent model for the rescue agents: fire brigades, ambulances and police force, as well as their operation centers; we also devised a simple model of civilians by allowing them to attempt to reach the closest refugee structure and run some test experiments. We finally discuss some preliminary ideas for carrying out systematic evaluation of situation assessment strategies, as well as more comprehensive evaluation criteria taking into account the overall performance of the system.

3. Prospects for future research and project exploitation

As shown above, the objectives of the project have been successfully achieved. This not only provides evidence that the proposed approach has proved successful, but also that there are interesting developments that can be foreseen. We first address the topics for future research and then briefly discuss the exploitation of the project results.

A major outcome of the project is the platform for evaluating the strategies for operating in a multi agent scenario where a major task is acquiring and maintaining a coherent representation of the world in the face of a partially known and rapidly evolving scenario and with respect to achieving specific goals within such scenario. In particular, the following are issues that may be effectively addressed in a new research project:

- *metrics for evaluation of situation assessment and situation awareness.*
- *analysis of strategies for situation assessment*
- *integration with (electronic) sensory input and robotic agents.*

The project is a first step towards a medium term goal of providing tools for supporting crisis management, to be deployed before hand, by providing both for qualitative and for quantitative analyses of the emergency plans and to be used during the emergency supporting monitoring and situation assessment. In this respect, the results of the project as well as some of the techniques can be exploited within the PRET 5 years program (Prof. Llinas, SUNY Buffalo).

The project can also be seen as a contribution to the goal of monitoring and coordinating the activities of a group of agents that can be human, human-operated and fully autonomous, which are operating in an emergency scenario in urban areas. In this respect the proposed solution, and, more generally, the research should be generalized in two respects:

- *integration with sensor infrastructure*
- *type of emergency/operation*

4. Publications

Below we list the publications appeared since the beginning of the project that are related to the field of the present project.

G. Grisetti, L. Iocchi, D. Nardi, Global Hough Localization for Mobile Robots in Polygonal Environments. In Proc. of International Conference on Robotics and Automation (ICRA2002).

L. Iocchi, D. Nardi, Hough localization for mobile robots in polygonal environments. To appear in Robotics and Autonomous Systems.

C. Castelpietra, A. Guidotti, L. Iocchi, D. Nardi, R. Rosati, Design and implementation of cognitive soccer robots. Proc. of RoboCup Symposium 2001.

C. Candea, H. Hu, L. Iocchi, D. Nardi, M. Piaggio, Coordination in multi-agent RoboCup teams, Robotics and Autonomous Systems, 36, pp. 67-86, 2001.

L. Iocchi, D. Nardi, M. Salerno, In Balancing Reactivity and Social Deliberation. Reactivity and Deliberation: A Survey on Multi-Robot Systems. In Multi-Agent Systems, (LNAI 2103) M. Hannebauer, J. Wendler, E. Pagello Eds. Springer, 2001.

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati, Data integration in data warehousing. Int. J. of Cooperative Information Systems, 10(3):237-271, 2001.

F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, Coordination in dynamic environments with constraints on resources. In AAAI-02 Workshop on Cognitive Robotics, July 2002.

F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, Generation and execution of partially correct plans in dynamic environments. In IROS-02 Workshop on Coperative Robotics, October 2002.

5. Attached Documents

[1] A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, M. Salerno, Structure and working of the rescue simulator.

[2] A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, M. Salerno, Information fusion.

[3] F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, Architecture and agent modelling for situation assessment.

[4] F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, Adaptability of the RoboCup Rescue Simulator.

[5] F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, Monitoring and information fusion for search and rescue operations in large-scale disasters, In *Proc. of Information Fusion (IF 2002)*, AnnaPolis, July 2002*.*

Adaptability of the RoboCup-Rescue Simulator

F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, and D. Nardi

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113 - 00198 Roma, Italy
{fdagosti,last_name}@dis.uniroma1.it,

Abstract. In this document we describe the tools we developed to aid the agent designer defining agents and to set up the simulation environment of a real disaster in the RoboCup-Rescue simulator. In addition, we discuss the use of these tools in designing a real rescue scenario.

The tools include the GIS editor, that allows one to build a map of the Rescue domain and the Agent Development Kit (ADK) for the Rescue domain, that is a specialization of the general ADK described in [3]. The GIS editor is a graphical tools supporting the specifications of a Rescue map by relying on geographic data of the choosen scenario when available. The editor outputs the map in a format that is suitable for use in the Rescue simulator.

The ADK is a tool suite supporting the design of agents. At this stage it allows to draw agent plans, using the Plan Assistant interactive tool, and to specify in a relatively easy way the other basic features of the agents in the Rescue domain, in particular, the primitive actions and the fusion strategy.

The GIS editor has been used in reconstructing a simulation scenario of the city of Foligno related to the earthquake of Marche and Umbria in 1997. In addition, we applied the ADK to design some elementary agent model for the rescue agents: fire brigades, ambulances and police forces, as well as their operation centers; we also devised a simple model of civilians and run some test experiments.

The document is structured as follows. First, we briefly describe the GIS editor. Then, we present the ADK for the Rescue domain: we first address its implementation on top of the RoboCup-Rescue ADK [1]; then we address the implementation of ADK agents in the RoboCup-Rescue domain. Both these sections contain notions that are needed to develop agents based on the ADK in the RoboCup-Rescue domain. The subsequent sections illustrate various aspects of the adaptation of the RoboCup-Rescue simulator to a new scenario: we start by sketching a simple model of the rescue agents, then we describe the scenario.

We finally discuss some preliminary ideas for carrying out systematic evaluation of situation assessment strategies, as well as more comprehensive evaluation criteria taking into account the overall performance of the system.

1 GIS Editor

Before describing the functionalities of the GIS editor, we briefly present the formats used by the RoboCup-Rescue simulator to store the GIS data and record the initial disposition of agents, fire-points, damaged roads/buildings, etc. The files containing the data for the RoboCup-Rescue simulator can be easily created/edited using our GIS-Editor.

GIS data are stored into several files:

- *Building.bin, Road.bin , Node.bin* - These files contain all the information about buildings, roads and nodes, respectively;
- *Galpolydata.dat, Shindopolydata.dat* - They store informations about the initial effects of the earthquake over buildings and roads, respectively (see [7] for details).
- *gisini.txt* - This is a text file containing the initial disposition of all agents and ignition points (initial locations where the fire propagates from) on the map.

The GIS editor is an interactive graphic tool for editing the GIS files: it allows one to create a RoboCup-Rescue map without any programming skills. The basic operations that can be carried out are:

- loading a raster file to draw the gis map on (this allow to manually creating a map from a bitmap picture)
- adding/modifying objects (nodes, roads, buildings) and object properties;
- specifying shindopolydata/galpolydata information, that controls the behaviour of the collapse simulator;
- setting the initial position of the agents in the simulated world;
- setting the position of the ignition points;
- transforming the world in various ways (mirroring/shifting/resizing);
- exporting/importing GIS information to/from a relational database for easy in-depth viewing/editing from other applications;
- checking a large set of map inconsistencies: in particular, it verifies if the graph formed by buildings, nodes and roads is connected, i.e. if it is possible to reach every location on the map from every other one.

The GIS editor can be downloaded at

`http://www.dis.uniroma1.it/~rescue`

For more information on the use of the editor please refer to the online documentation. This tool has been used in the modeling of the city of Foligno, whose model is presented in a later section.

2 ADK and the RoboCup-Rescue Domain

In this section we describe the RoboCup-Rescue world model and our implementation of ADK in this particular domain (refer to [3] for the notions concerning

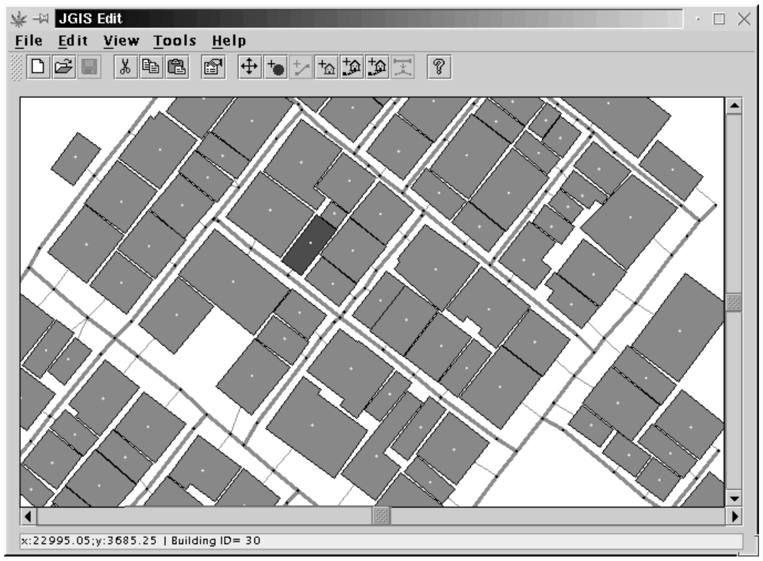

JGIS Edit
File Edit View Tools Help
x:22995.05;y:3685.25 | Building ID= 30

Fig. 1. JGisEditor Interface

the general framework).

Our work exploits and extends the RoboCup-Rescue Agent Developement Kit, made by Micheal Bowling (see [1]) that is a general programming-framework for implementing agents for the RoboCup-Rescue simulator. That tool provides some basic mechanisms for exchanging data with the RoboCup-Rescue Simuator, as well as a set of primitives that hide the low level implementation details to the agent developer. We specialized the Bowling's RoboCup-Rescue ADK, by adding a set of modules and functionalities, in order to implement the architecture described in [3]. In the following, the term *ADK* or *Extended ADK* refers to our agent developement tool, while *RoboCup-Rescue ADK* denotes the one discussed in [1]. We first describe the RoboCup-Rescue World, then we discuss the adaption of the Rescue structures to cope with the features required by the ADK modules, and the communication primitives we added to the original RoboCup-Rescue ADK agent capabilities. Finally, we discuss agent state monitoring features added to analyze coordination and fusion strategies.

2.1 RoboCup Rescue world model

The RoboCup-Rescue world model is minimal, but it could be easily extended to fit real scenarios more closely. It deals with three main entities or *object classes*: *buildings*, *roads* and *nodes*, respectively. *Building* objects represent every kind of building on the map: houses, police offices, hospitals, fire stations, ambulance centers, refuges, etc. As a result of a earthquake shock, a building can collapse and obstruct a road; moreover, a building can catch fire more or less likely, according to its construction material; for example, a concrete building is less flammable than a wooden one. Further, buildings can have one or more floors and one or more linkage points with the surrounding roads. A *Road* can be partially or totally obstructed by rubble in consequence of the collapsing of an adjacent building. Further, a road has one or more traffic lanes on each side and can have a sidewalk or not. The road network is described by a graph having one or more edges for each road and one node for each crossroad and for each junction between adjacent edges constituting a road.

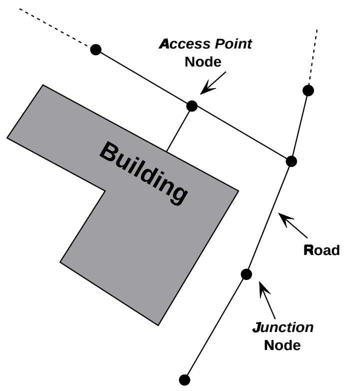


Fig. 2. GIS Objects

Each object class (*building*, *road*, *node*) is characterized by a number of attributes describing a specific instance of the class. The main attributes are the following:

- Buildings : *Plant*, *Kind*, *Material*, *Fieryness*, *Brokenness*, *Floors*, *Access-points.*
- Roads : *Kind*, *Length*, *Width*, *Block*, *Repair-Cost*, *Lines-to-head/Lines-to-tail*, *Sidewalk-width.*
- Nodes : *Connected-Roads*, *Signal*, *Signal-timing.*

2.2 Interfacing ADK and RoboCup-Rescue Simulator

Using the ADK in developing a multi agent system that works in the RoboCup-Rescue Domain, one has to face two main issues:

- the extension RoboCup-Rescue World, for fitting the ADK World Requirements
- the interface between the ADK communication subsystem and the communication primitives provided by the simulator

Before addressing them, we recall that in the RoboCup-Rescue Agent Development Kit [1], two main classes are defined:

- a `Controller` class, that is the base class of the user agents. In such a class the basic operations of the agents, like moving to a location, extinguish a fire or communicate, are defined.
  An ADK Agent is built on the top of such a class, and extends the interface, while keeping compatibility. The major improvements are the definition of a communication interface that allows channels, an interface to the fusion modules, and a plan execution module.
- a `Memory` class that implements the world structure. Such a memory class has been extended in order to fit the requirements of the ADK. The major features added are:
  - module introspection
  - serializability of the object properties over the agent communication channel.

World Objects The RoboCup-Rescue Objects fit a good part of the above discussed requirements about the ADK World Objects, in particular they provide, for each object: serializability and uniqueness (all the agents know the same object with the same key). We only need to extend their definition in order to add the concept of Property Information Source (Justification) and some kind of introspection capability. Both features are needed for developing the information fusion component. The definition of an object as a set of properties, instead of an already-built, well known class, leads to the design of platform independent fusion strategies. However, the possibility of seeing the world as a Rescue World is not lost, since it is useful for testing special domain dependent strategies.

We also manipulate the way an agent receives data from the simulator, to be able to implement the state reconstruction schema discussed in [3]. In the RoboCup-Rescue ADK the data coming from the simulator are directly written into the agent memory. Since we want the Simulator updates as well as the agent Reports to be stored in the Sensor Memory, we need to modify that part of the data path.

Moreover, in order to maintain backward compatibility with the RoboCup-Rescue World and with the RoboCup-Rescue ADK, we write our data in a structure derived from the original one and we need to remap the original access functions. In Figure 3 the data path modifications are shown.

Communication and Coordination The Rescue simulator allows an agent to communicate with the others through two medias: voice and radio. Only the rescuers (police agents, fire brigade, ambulance team) can use the radio. An agent using the radio can communicate, in a broadcast way, only to his central station and to agents of the same kind (i.e. a Police Agent can only communicate with police agents and police station). The central stations can communicate with each other, although they are a different kind of agent. Since we want to define

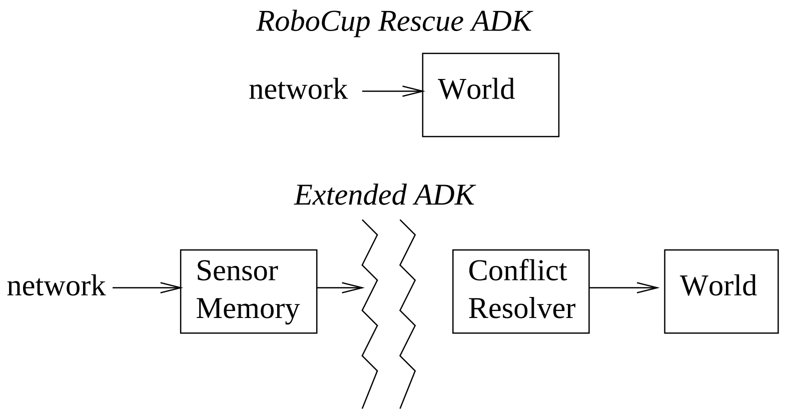


Fig. 3. Changes in the Rescue Agent Memory Update Path

groups of communicating agents, we assign a radio channel to each group. The Rescue Framework allows the transmission of text messages and we have defined a simple message format that packs the channel number in the message body. On the receiver side, an agent simply discards the messages having a channel number different from their own. Summarizing, the communication operations an agent can perform are the following:

- telling/hearing a message through the radio on the selected channel;
- setting/quering the radio channel;
- hearing a radio message from a generic channel.

The coordination of the Rescue Agents, as described in [3], makes use of two kinds of coordination messages: the *commands* from the central station to the agents acting in the environment and the *intentions* for dynamic task assignment. These messages are distributed on virtual networks simulating broadcast radio communication as described above and, due to the publish/subscribe mechanism, they are received only by the agents that are entitled.

2.3 Monitoring the fusion process

As described above, each agent stores its knowledge about the world in his memory (the "standard" memory or the "extended" Sensor Memory), that is a set of objects and object properties, and build a representation of the world based on various data recorded in its memory. Thanks to the ability to communicate by voice or radio, agents can exchange information; consequently, at each simulation cycle, the agent memory is updated, resulting in updates of the world representation with new objects, or object properties with new values. It is interesting for analyzing situation assessment to follow the evolution of agents' knowledge during the simulation progress: the world knowledge of agents developed using the ADK framework can be monitored through a generic Robocup-Rescue viewe. In fact, at every time it is possible to connect a viewer module to an agent module allowing to view the agent world knowledge and follow their changes during the simulation.

3 Implementing agents using the ADK

In this section we shortly present the Plan Assistant, which can be used to design the agent plans and interface and the functions for defining primitive actions and fusion capabilities. The latter are not intended to be a complete documentation, but a short introduction to help specifying the code that implements the agents. They require some knowledge of the RoboCup-Rescue Agent Developement Kit Manual [1], and can be skipped by the reader not interested in implementation details.

3.1 Plan Assistant

Agent plans can be represented as graphs; the Plan Assistant is an interactive tool for designing plans by drawing their representative graphs: it allows to place nodes and connect them with edges. Moreover, both edges and nodes can be labeled with information. The Plan Assistant interface is shown in Figure 4. The basic operations allowed by the Plan Assistant are:

- add/delete/label a node
- add/delete/label an edge
- load/save a plan to/from a plan file
- save a file in a printable format

Since the graph representing a plan must be coherent with the plan representation, as specified in [3], a rule check mechanism has been added, in order to verify some properties of the drawn plan; as an instance, the program does not allow to save a plan where a goal node has not been defined. Moreover, there are some rules on the kind of edges leaving from a node: at most one edge can contain an ordinary action; if a sense action *true* branch leaves from a node, also the *false* branch has to leave from that node.

For a detailed explanation of the plan representation see [3].

3.2 Action Implementation

The implementation of primitive actions requires the following definitions:

- an extended set of operations
- action interface
- a procedure for loading the plans from a file

.

In order to implement an action the user has to extend one of the following two classes: `pemAction` or `pemCondition`, if the action being implemented is an ordinary action or a sensing action, respectively. Before presenting the interface for defining primitive actions, we recall the Basic rescue operations ad Communications operations.

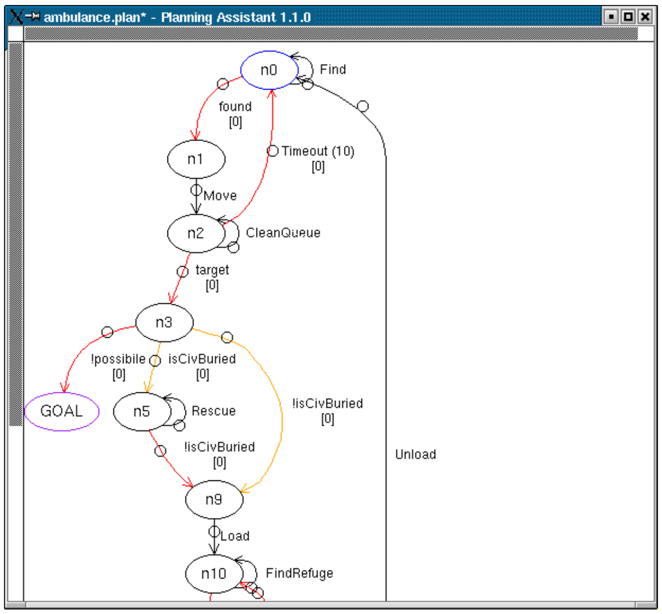


Fig. 4. Plan Assistant

Rescue Basic Operations The basic operations (see [1]) can be summarized as follow:

- `void act_move(Object **path)`: move the agent along the specified path;
- `void act_open(Road *target)`: remove debries from a blocked road.
- `void act_rescue(Humanoid *target)`: rescue an injuried civilian;
- `void act_load(Humanoid *target)`: load an injuried civilian on a car;
- `void act_unload()`: unload a previously loaded civilian;
- `void act_extinguish_simple(Building *target)`: extinguish flames in a burning building.

Communication Operations In addition to the above, our ADK introduce the following communication operations:

- `void act_radio_tell( const char* msg)`: tells a message through the radio on the selected channel
- `void setChannel(int)`: sets the radio channel
- `void getChannel() const`: returns the radio channel
- `virtual void radio_hear(const char* msg)`: called by the system whenever an external message sent to the current channel is received
- `virtual void radio_hear(int& chn, const char* msg)`: called by the system whenever an external message is sent without doing channel check. Useful for implementing central coordination agent that has to hear concurrently on more than one channel.

Through these primitives we allow for the definition of groups of agents belonging to the same radio channel. The communication within each group is broadcast. There are some limitations in the creation of groups: it is not possible to create groups of agents having different type, and an agent can communicate only with its central station. Central stations can communicate with each other.

Ordinary Action Interface (`pemAction class`) An ordinary actions implements the following interface:

– the `void Initialization()` method, that is called whenever the plan scheduler enters in a new state;
– the `void Termination()` termination method, that is called whenever the plan scheduler leaves the state;
– the `void ExecuteStep()` method, that is called each scheduler cycle;
– the `ActionState getActionState()` method, that is invoked by the scheduler to check the action execution status. An action can be in-progress or finished.

Sensing Action Interface (`pemCondition class`) A sensing action implements the following interface:

– the `void Initialization()` method;
– the `void Termination()` termination method;
– the `bool Eval()` method, that is called each scheduler cycle to test the value of the sensing action.

Loading Plans and Actions After loading a plan from a file the instantiated actions are expressed as character strings, that need to be bound with the corresponding executable action class instances. This is done by the `ActionManager` class that allows one to bind an ordinary action and a sensing action, respectively, through the `void addAction(..)` and the `void addCond(..)` methods. A different instance of the `ActionManager` class is needed for each agent instance.

After loading a plan through the `ActionManager::planLoad(const char * filename)` method, it is possibile to refer to the loaded plan as an action using the `ActionManager::getAction(const char* actionName)`. Since a plan is seen as a simple action it is also associated to a string key. Such a string key is the value of the *filename* argument of the loading function. That allows for the definition of hierarchical plans by simply labeling higher level plan actions with the names of the lower level plans, then iteratively loading all plans and base actions.

3.3 Information Fusion Implementation

As previously said, ADK implements an information fusion module. Here we discuss the steps needed to use and extend such a module. Since the fusion module is implemented as an extension of the `Controller` and `Memory` classes provided by the RoboCup-Rescue ADK [1], here we describe the extensions and the enhancements made to such classes.

In order to activate the information fusion extension, the user has to execute the following steps:

– Activate the world extensions, such as introspection, that are needed by the fusion processes.
– Design a fusion policy by choosing:
  - how to solve the conflicts;
  - when to send a *report*[1].

First we deal with the structures that have to be instantiated by the agent classes, then we look at the specific task of each overridable method. Moreover, we show what kind of initialization has to be done in order to activate the fusion process. Finally, we trace the execution of the fusion process through the sources, in order to highlight the execution context of each of the methods that has to be redefined in customizing the fusion strategy.

Agent Class: Fusion Structure Instantiation For each agent is needed:

– A `Controller` derived class, that implements the agent specific methods, by overriding the original virtual methods.
– A `Memory` or a `Memory` derived class, that implements the agent memory.
– A `SensorMemory` or a `SensorMemory` derived class.

These classes have to know each other. This is easily done by the addition of the following rows in the Controller derived class constructor.

```
MyAgent::MyAgent(int type) : Controller(type, new Memory)
{
m_memory = (Memory *) Controller::m_memory;

/*make the memory to know the owning controller*/
m_memory->setController(this);

/*Sensor Memory creation*/
sm=new SensorMemory();

/*make the Memory to know the SensorMemory*/
m_memory->setSensorMemory(sm);

/* rest of the constructor ..*/
```

[1] the atomic information unit (see [3])

Implementation of a fusion strategy In order to implement a fusion strategy a user has to:

- define a Conflict Resolution strategy by implementing an ad-hoc class or by the use of the class
  `SimpleConflictResolver`

- define what to do when a report is received through the network by eventually redefining the method
  `Controller::parse_update(const char*)`

- define what to do once an object changes its state by redefining the method
  `Controller::sense_change(Object *o)`

- define which parts to send of each updated object by redefining the method
  `Controller::send_sensed_object(Object *o)`

Program and Agent Initialization When the program starts, the static structures for the introspection of the world objects have to be set up. To this end, at the beginning of the `main()` method, the following must be added

```
objdef_init();
```

After the structures used by the fusion modules have been instantiated, the activation of the fusion process can be done by calling, after the connection and initialization stages of the Agent, the method
`SensorMemory::setActive(bool)`

After the initialization stage the virtual method
`Controller::init()`
is called, so to enable the fusion it is just needed to override it with a function that, in its body, invokes the
`SensorMemory::setActive(bool)`
on the Sensor Memory of the agent, with a `true` argument.

Execution Trace At each cycle the fusion process procedes as follows:

1. The agent sensing is broken into many reports, that are stored into the Sensor Memory, with Justification value meaning the agent senses.
2. The external reports (by other agents) are collected in the Sensor Memory. Each time a new report arrives, it has to be parsed and stored, via the method
   `Controller::parse_update(const char*)`

3. The virtual method
   `Controller::integrate_changes()`
   is called; this starts the analysis of the collected data. A `ConflictDetector` class finds the conflicts identified in the current time of the sensor memory; then a `ConflictResolver` class performs the fusion process.
4. The `Memory::integrate_changes()` method is called on the output of the `integrate_changes()` in order to modify the agent update process; this causes the updates to be partitioned by the Object Identifier, in order to cluster the changes that affect each object.
5. For each modified object the method
   ```
   Controller::sense_object(Object* o,
      const list<ValueReportMap::const_iterator>& propertySet)
   ```
   is called. This is a virtual method, so the user can override it.
   The agent memory update is performed only by the method
   `Controller::sense_object(...)`.
   If this method is redefined by the user, a call to the original method has to be done within the redefined method body in order to affect the agent knowledge. That means that *prior* to call the method
   `Controller::sense_object(...)` the object is unmodified, while after it is.
6. After an object is updated, the empty virtual method
   `Controller::sense_change(Object *o)`
   is called.
7. The virtual method
   `Controller::send_sensed_object(Object *o)`
   is called, in order to send a message with the updated world objects. Here the user can specify *when* a report about an object has to be sent.

4 Modeling Rescue Agents

In this section we sketch a simplified model of agents that can be used for a first set of test experiments. The definition of refined models for agents is beyond the scope of the present work. There are two categories of agents in the Rescue domain: the ones that have to be rescued and the rescuers. Civilians belong to the first category, while police forces, ambulances and fire agents belong to the second one.

The task as well as the operational capabilities of each agent depend on its category. As an instance, an Ambulance Team cannot extinguish a burning building. The role of each category of agents with rescue capability is the following:

– Police Team: free the road blocked by debries.
– Fire Team : extinguish fires.
– Ambulance Team: rescue injuried people.

For each of the rescue agent categories, a central station is defined, that is modeled as an agent, whose goal is to coordinate other agents' operations.

While agents of different categories cannot communicate, central stations can communicate with other stations.

In order to simplify the framework for testing, we made some design choices:

– Situation Assessment is performed only by the central stations, via the reports received by the rescue agents.
– The agents execute the tasks given by the central station they belong to, and send reports about the sensed world.

4.1 Police Force

In the Rescue domain, the role of the police force is to free the roads blocked by the debries in order to allow civilians and rescuers to pass through them.

A police force agent can belong to one of these two categories: *Police Agent*, which models a human agent who directly acts in the world, coordinating its behaviour with teammate through distributed coordination; *Police Center*: agent that performs centralized task allocation and state reconstruction.

Police Agent The PoliceAgent is designed according the following criteria:

– A police agent can be in one of these two states: BUSY or IDLE, depending on its current task allocation.
– A police agent knows only the information percepted by his sensors, plus (optionally) the information sent by the police center concerning the area where to operate.

An agent is in the idle state by default. It switches in busy state once allocated by the police center to one of the following tasks:

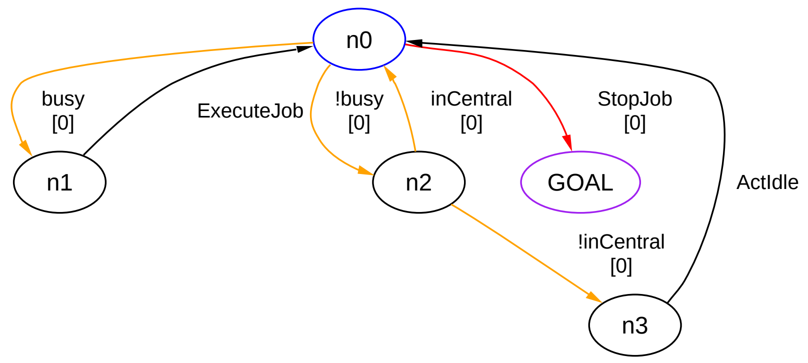
– to clean a road or an area (commandClean)
– to explore an area (commandGo)

When the tasks have been performed, the police agent switches back in idle state.

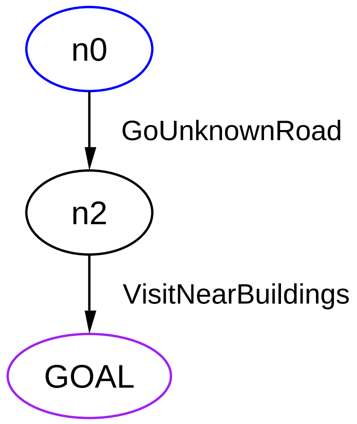
A police agent in idle state can be within the central (inCentral) or outside. If it is in the central, it waits for allocation, while, if outside it explores the environment by entering unknown buildings (VisitNearBuilding) or checking roads(GoUnknownRoad).

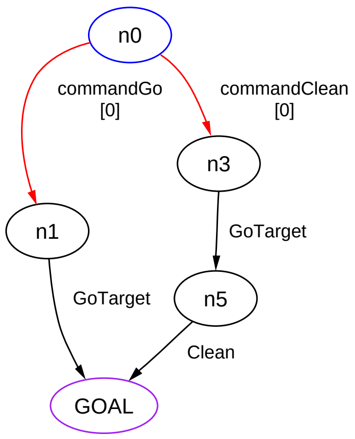
Figure 5 presents the plan executed by a Police Agent; light (red) arrows correspond to sensing actions, dark (black) arrows correspond to ordinary actions.

PoliceCenter It performs resource allocation to assign task to the agents using a simple method, and integrates the information coming from the police agents exploring the environment. Once received a fire notification, it sends the PoliceAgent that is in idle state and is nearest to the area affected by the fire. In case the Police Center receives a notification of an injuried victim, it does the same as for fire notification, sending the nearest free police agent; in addition, a request to the ambulance center is made.



Main Plan





ActIdle Plan ExecuteJob Plan

Fig. 5. Plan executed by the Police Agent

4.2 Fire Team

Fire Team agents can belong to the following categories: the *Fire Agent*, that model human agents that operate in the real world to extinguish fires, and the *Fire Center* which is a coordinator agent.

Fire Agent A Fire Agent can search a fiering area or move to an area specified by the Fire Station. When it is close to the fire, it tries to extinguishing it. It stops the extinguish action and starts a new search when one of the following situations occurs: the fire has been extinguished, or the building has been completely destroyed.

Fire Station The fire station collects and integrates the information sent by the fire agent, and allocates Fire Agents with a simple policy as the Police Center.

4.3 Ambulance Team

An Ambulance Team agent can be of two kinds: `Ambulance Team` or `Ambulance Center`. The first is a model of the human agent that has to extract civilians and agents from the debries of the collapsed buildings and carries them to the refuges, while the second one takes care of allocating the resources using his world knowledge.

Ambulance As for the Fire Agents, Ambulance teams can search injuried people or move to a location specified by the Ambulance Center. When close to a victim, the Ambulance loads it and brings it to the nearest refuge.

Ambulance Center The ambulance center collects and integrates the information sent by the ambulance team agents, as well as requests from other centers. Like the other rescue centers, it performs a simple task allocation.

4.4 Civilians

Their basic behaviour of the civilians is to go in the refuges once the alert situation begins. Optionally they send help messages that can be heard by the central stations or the neighbour rescue agents.

5 Modeling a real scenario

In this section we show how to set up a RoboCup-Rescue simulation starting from a real disaster scenario, which requires the following steps:

- Developing a set of rescue agents having the desired behaviours; this involves the use of the ADK for developing the agents, and the Plan Assistant for defining the agent plans.
- Designing a map for the choosen site. This could be done using the GIS Editor.
- Setting the initial position on the map of agents, rescue centers and ignition points, also done with the GIS editor.
- Defining a proper execution asset.

While the definition of rescue agents was given in the previous section, below we address the other steps.

5.1 The Earthquake of Umbria and Marche

Throughout the fall of 1997, a serious earthquake affected the Italian regions of Marche and Umbria: many housing estates as well as important monuments were heavily damaged, first and foremost the world-famous Basilica of S. Francesco in Assisi. In order to experiment and verify techniques and methodologies developed in our work, we have selected Foligno, one of the most important cities in that region, as a scenario for significant disaster simulations in order to test the applicability of the approach; below, we discuss the features of the chosen site and describe the main aspects of its representation within the simulator.

5.2 Domain features

Foligno is located in a flat region of eastern Umbria. Its urban structure is characterized by a medieval center surrounded by more recent suburbs; in particular we are focusing our attention on an area of about 4 km^2 corresponding to the city center.

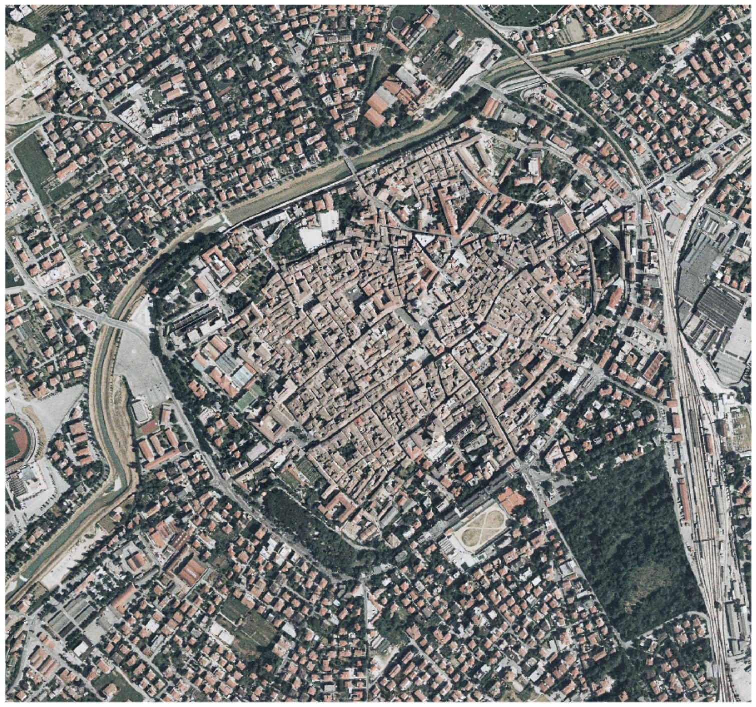


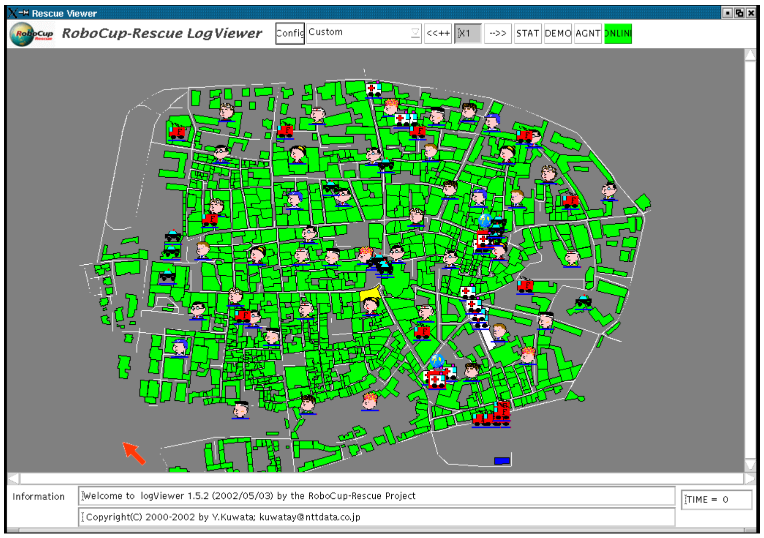
Fig. 6. The center of Foligno

In the area under consideration there are no high-rise buildings; most recent structures are mid-rise, in the four- to nine- story range. All large, multi-story buildings were constructed of reinforced concrete. Oldest buildings were mainly constructed of rubble-work, whereas only few structures are steel frame buildings or wood buildings. There are no industrial structures; most buildings are housing estates having variously-shaped plants. The road network is quite irregular, with not very large roads and narrow alleys.

Important public estates are situated inside this zone: the Hospital, the Fire Station, the main Police Center as well as other less important Police offices. Many buildings in this area are very interesting from the hystorical point of view: most of them are well-preserved Middle-Age-dated estates. The earthquake injured many of them quite seriously.

From our point of view, the choice of Foligno as the scenario for our simulations is very suitable for several reasons:

- It is a not very large town, but it has nevertheless all the items we consider most interesting for the Robocup-Rescue simulator: the hospital, the fire station, the police office, etc. etc.
- Quite a lot of information about this site (maps, data on buildings and roads, etc) is available.
- The results obtained during our experiments with the simulator (best intervention strategies, optimal resources allocation, etc.) can be analyzed taking into account the (well-documented) behaviours of the real (human) rescue forces immediately after the earthquake.



Welcome to logViewer 1.5.2 (2002/05/03) by the RoboCup-Rescue Project

Copyright(C) 2000-2002 by Y.Kuwata; kuwatay@nttdata.co.jp

TIME = 0

Fig. 7. Foligno Map seen by the Simulator

5.3 Initial conditions

The simulator initial conditions consist of:

- the initial position of each agent;
- damage of each building in the map;
- the initial fire points;
- the collapse simulator initial conditions.

In order to model the initial condition we placed the civilians according to the people distribution over the territory, taking into account the function of each building. Such a distribution has high values in buildings of public use, low values in poorly populated areas.

In the initial conditions we modeled we have:

- 50 groups of civilians
- 12 fire teams
- 12 police forces
- 12 ambulances

plus a fire station, a police office, an ambulance center and an hospital (the refuge).

The collapse simulator initial conditions has been modeled taking into account a feasible earthquake center. The fire points can be placed in the map to maximize the simulation dynamics.

5.4 Execution Asset

Due to the high level of processing power requested for executing the simulation, we used a cluster of four Linux based boxes connected by an high speed network (for the details and configuration, see [4]). The processes are distributed on each machine in the following way:

1. kernel, miscellaneous simulator, road block simulator, 10 civilians
2. traffic simulator, fire simulator, collapse simulator, 10 civilians
3. police agents, police office, fire station
4. fire agents, ambulance agents, ambulance center

to uniformly distribute the computational load in order to maximize the performance.

6 Evaluation of Situation Assessment: preliminary remarks

An interesting possibility offered by the RoboCup Rescue simulator is the evaluation of different techniques for information fusion and situation assessment. Such an evaluation could rely on measures that can be effectively computed through simulations.

Specifically, the evaluation of situation assessment can address the following aspects:

– effectiveness of a particular fusion strategy;
– robustness with respect to errors in sensing, and network noise;
– certainty on the reconstructed situation;
– computational effort.

Situation assessment can be measured by comparing the real-world as known by the simulation framework and the world known by the agent who performs situation assessment. Moreover, the overall system performance, as well as the rescue policy strategies, can be measured in terms of the number of rescued agents.

Errors can be artificially introduced to simulate sensor or network noise, to verify how they affect the performance. As an example, the following experimental settings can be considered in order to analyze situation assessment by the central stations:

– *Error-free*: no errors in communication and no erroneous indications by the agents. In such a setting, the overall system load can be evaluated, as well as the ideal performance of a fusion technique.
– *Erratic Agents*: they provide possibly wrong information. By introducing agents sending incorrect estimation of the world it is possible to have a measure of the robustness of the fusion process, with respect to the noise in the information reports.
– *Communication Noise*: messages can be lost. The introduction of network noise, in terms of undelivered messages, can give an estimation of the sensitivity of the information fusion process to communication failures.

In addition to the specific fusion technique, there are several other factors that can influence the behaviour of the system, and, consequently, the evaluation of situation assessment; among them:

– the agent behaviours;
– the resource allocation policies used by the central stations in assigning tasks to the agents;
– the coordination strategy used among agents at the same level.

A complete evaluation of situation assessment should address also these parameters, because not only the overall performance, but also the knowledge acquired by the agents can be dependent on them. Therefore, it is an interesting and challenging task to define a proper set of experiments that can take into account of such a broader view of situation assessment.

References

1. Michael Bowling. Robocup rescue: Agent developement kit. *http://robomec.cs.kobe-u.ac.jp/robocup-rescue*, 2000.
2. Robocup Rescue Technical Committe. Robocup rescue simulator manual v0.4. *http://robomec.cs.kobe-u.ac.jp/robocup-rescue*, 2000.

3. F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, and D. Nardi. Architecture and agent modeling for situation assessment. *Technical Report*, 2002.
4. F., A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, and M. Salerno. Structure and working of the rescue simulator. *Technical Report*, 2002.
5. A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, and Salerno M. Information fusion. *Technical Report*, 2002.
6. D. L. Hall and J. Llinas. *Handbook of multisensor data fusion.* CRC Press, 2001.
7. T. Takahashi. Tools for checking and creating ***polydata.dat files. 2002.

Architecture and Agent Modeling for Situation Assessment

F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, and D. Nardi

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113 - 00198 Roma, Italy
{fdagosti,last_name}@dis.uniroma1.it,

Abstract. In this document we deal with multi agent systems for "Situation Assessment". Situation Assessment is the task of reconstructing the knowledge concerning a physical scenario that is dynamically and unpredictably changing, as, for example, in the case of natural disaster. In particular, we aim at assessing the situation in the framework of a simulation scenario, where a variety of external events can occur and a number of agents are in charge of handling the emergency, by accomplishing a specific mission, such as human life saving. Information about the situation can be acquired in several ways, through inspection by the agents (rescue parties) or through a variety of external sources. A precise understanding of the situation is fundamental to foresee its possible evolutions and to decide the actions to be taken.
The aim of the work reported here is to define the architecture and the basic features of the the software agents that are needed in order to properly address the task of Situation Assessment. The intended application domain is the RoboCup-Rescue Simulator, which provides a suitable basis for developing and testing multi agent systems with the above described features.
The architecture of our multi agent system thus combines the requirements of deliberation and reactivity, that have been pursued by through the design of so-called cognitive agents, together with the ability to coordinate the behaviour of several agents that contribute to the assessment of the situation, while fulfilling their mission. More specifically, under the assumption that the agent can communicate according to some structure, possibly referring to different organizations, we model the basic features that characterize the behavior of an agent: the plans that allow the agent to accomplish its task, its information fusion policies, its capability to cooperate with other agents.
The document is structured as follows. In the first section we describe the basic features of cognitive agents and discuss the main design choices and functions of a system of cooperating agents. We then present the basic features of the Agent Development Kit (ADK), which provides a basis for the design of agents that are capable of executing plans, combine information coming from several sources and cooperate in order to accomplish a common goal. Subsequently, we present the main features of the implementation of our ADK, specifically addressing the plan specification and the information fusion component.

1 Agent Modeling

The notion of autonomous agents is central to the field of Artificial Intelligence and the design of intelligent agents (both physical agents, i.e. robots, and software agents) is one of its foundational research goals.

Initially, the focus of the research on the design of autonomous agents has been on the high-level representation of actions that they can perform (see [30] for an historical perspective). However, it soon became clear that it is very difficult to build agents that exhibit the desired behaviour in real environments, simply on the basis of such a declarative representation. Consequently, research split into two streams, that developed rather independently of each other. On the one hand, the basic functionalities of the agents have been developed (see for example [3] for a discussion of reactive architectures for mobile robots); on the other hand, an agent needs not only the ability to promptly react and adjust its behaviour based on the information acquired through its sensors, but also to achieve high-level goals. Therefore, it should also be able to reason about the actions it can perform, find plans that allow it to achieve its goals and check whether the execution of the actions leads to the accomplishment of the goals. The integration of reactive and planning capabilities has thus become a focus of research in mobile robotics (see for example [31, 15, 33]).

We believe that this renewed effort to combine a logic-based view of the agent with its reactive functionalities is essential to devise agents that operate in a real world environment. To this end a new research field has been developing in the last years. It is named *Cognitive Robotics* [25] and it aims at designing and realize agents (mobile robots as well as simulated agents operating in virtual representations of real environments) that are able to accomplish complex tasks in real, and hence dynamic, unpredictable and incompletely known environments without human assistance, and that, to this purpose, can be controlled at a high level by providing them with a description of the world and expressing the tasks to be performed in the form of goals to be achieved.

The peculiar features of a *cognitive agent* are:

- the presence of cognitive capabilities for reasoning about the information sensed from the environment and about the actions it can perform;
- the ability of properly and promptly reacting to changes occurring in a real environment, that is *dynamic*, since changes can occur at every time and a timely response to these changes is sometimes a critical factor, *unpredictable*, so that the effect of changes in the environment cannot be always and completely foreseen; *partially known*, hence it is not possible to have complete information on the environment, since many situations cannot be known a priori.

In this section we first present an architecture for modeling and implementing cognitive agents that has been devised and actually adopted for realizing many actual cognitive robots ([21], soccer robots [4] and also cognitive agents for the information extraction from the Web [10]). Then, we address the problem of coordinating a team of such cognitive agents.

1.1 Simple Agent Model

In principle, an intelligent agent can be seen as a device capable to perform three kinds of activities:

- *sensing*: reading and interpreting the external world information to maintain an internal state of the world, representing the knowledge of the agent about the environment;
- *planning*: choosing which actions should be performed in order to accomplish a given task;
- *acting*: performing the chosen actions in the environment.

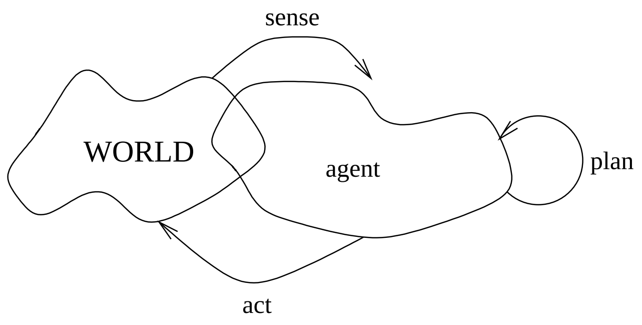


Fig. 1. Base Agent Model

A cognitive agent operating in a real dynamic environment must perform these activities periodically, implementing the so-called *sense-plan-act cycle*. However, in the actual implementation of the agents this solution is not feasible, since it does not take into account the fact that for complex tasks these activities may be very time consuming, thus affecting the reactivity of the agent in the real world [3].

This is the main reason for which many researchers have proposed different approaches for designing autonomous agents [3], often without an explicit representation of the knowledge of the agent and of the associated reasoning capabilities of the agents.

The solution adopted by most of the recent works in this field has been the development of hybrid architectures that allow for both the implementation of the *sense-plan-act cycle* and the definition of reactive behaviours in response to the environment changes. A classification of such hybrid architectures makes a distinction between those that make use of a single common representation for the information of the agent about the world [19, 33, 31] and those that use heterogeneous representation for these data [15].

The architecture presented in this section (see also [21] for further details) has three main features:

- *different layers*, that are useful for better modularizing the design and the implementation;
- *asynchronous modules*, that allow for integrating cognitive and reactive capabilities;
- *heterogeneity*, that allows for using different techniques in the different modules of the architecture.

1.2 System Architecture for a Cognitive Agent

The software architecture of our agent, shown in Fig. 2, presents a cognitive level, based on a logic representation of the agent's knowledge, and an operative level, based on a geometrical representation of the environment and a set of values denoting the internal state of the agent.

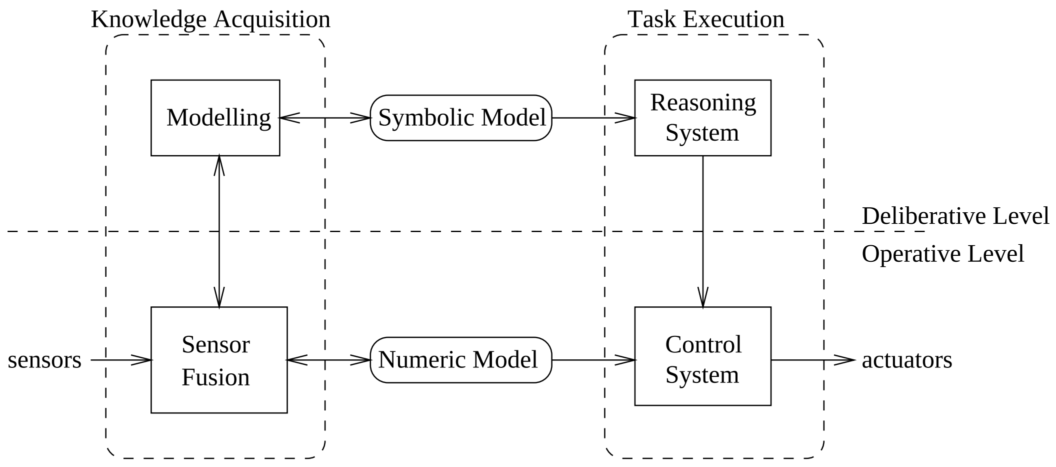


Fig. 2. Layered Agent Model

The high-level knowledge of the agent about the environment is constituted by a description of the dynamic system, containing information about the state of both the agent and the environment. This knowledge is represented over two levels: a deliberative model, including a symbolic knowledge base, and an operative numeric model, constituted by sensing data and numeric object properties.

Operative Level The operative level of an agent is very specific to the implementation domain and thus will be described in the next sections of this report.

Deliberative Level The deliberative level is mainly concerned with an explicit representation of the agent's knowledge, based on Description Logics. This knowledge is formed by both a general description of the environment provided by the agent's designer and the information acquired during task execution. The language specification and the reasoning services used in this level are described in the next subsection.

The deliberative level is formed by two components: a high-level knowledge acquisition module and a reasoning system.

The knowledge acquisition module is in charge of building and maintaining a knowledge base expressing a high-level model of the world in which the agent is embedded. This process makes use of artifacts in the numeric representation of the world, referring to objects in the environment, and the results of sensing behaviours to update the agent's knowledge about the environment. In this way, every high level atomic predicate can be monitored. For example, the property of being in a room is satisfied as long as the position of the agent is within the range of the artifact representing the room, while a door closed is detected by an interpretation routine driven by a sensing behaviour.

The reasoning system processes the knowledge base containing information on the world in order to make decisions about actions to be performed for goal achievement and a planning procedure has been developed in order to generate plans. A sketch of the main features of the representation and reasoning system is given below.

A Brief Overview of the Reasoning System Here we briefly recall a formalism for reasoning about actions, that has been recently proposed in [8, 22], and the implementation of a planner based on this formalism.

The main objective is to develop reasoning tools for a cognitive agent, and in particular a plan generation procedure for selecting the actions that will lead to the achievement of a given goal. Observe however that, because of incomplete information on the environment, achieving a goal usually depends on the possibility of performing actions for knowledge acquisition. So the reasoning system must be able to take decisions about the execution of both actions involving changes in the dynamic system, for example *moving actions*, and *sensing actions*, that are knowledge producing actions affecting only the knowledge of the agent about the system.

The basis of our proposal for reasoning about actions is provided by Propositional Dynamic Logics (PDLs) [22, 8], suitably extended with a nonmonotonic modal operator of minimal knowledge. In the standard framework, the dynamics of the system is specified in terms of *what is true in the world.* However, in a cognitive agent the dynamics is specified in terms of *what the agent knows of the world.* The idea is that the agent achieves its conclusions based on its *epistemic state* and not on the actual state of the world. This change of viewpoint in the representation has two important consequences: (i) it simplifies reasoning, and in particular it simplifies the task of deducing plans; ii) it makes deductive planning always constructive, i.e. the reachability of a state in which the goal holds is deduced only if there exists a known sequence of actions that leads to it.

Following the approach in [29], the behaviour of the dynamic system can be specified by means of a set of axioms that are described below.

- *Static axioms* specify properties which are true in every state and do not depend on actions. In other words, static axioms are used for representing background knowledge that is invariant with respect to the execution of actions.
- *Precondition axioms* specify circumstances under which it is possible to execute an action. We assume that the designer of the system is able to specify *sufficient conditions* for actions to be executed.
- *Effect axioms* specify (direct) *effects* of an action if executed under given circumstances, i.e. if executed in a state satisfying certain *premises*. Obviously, through static axioms, additional effects can be inferred from those specified by effect axioms. Observe that, in general, we do not require the designer to specify all the effects of an action.
- *Frame axioms* are used for expressing different forms of inertia laws. In particular, we are able to specify *default frame axioms*, i.e., default persistence rules which state that, if in the current state the property C holds, then, after the execution of the action R, the property C holds, if it is consistent with the effects of R. We also use *epistemic frame axioms* in our specification, which are able to express "causal" persistence rules. Such axioms are used to represent the fact that a property C is propagated only if another property D holds in the successor state. By suitably instantiating the above kinds of frame axioms in our system specification, we are able to formalize both inertial and non-inertial properties, and both inertial and non-inertial actions, thus addressing both the frame problem and the presence of exogenous events in our framework.

In addition to these kinds of axioms, we assume that the designer specifies the knowledge about the initial state, by providing an *initial state description* in terms of the properties (not involving actions) that are associated with the initial state.

Actions are assumed to be *deterministic*, which means that at most a single successor state is determined. However, the properties associated with such successor state are generally different in different interpretations. We might say that actions are deterministic but their effects are *underdetermined* in general.

The planning problem in our framework can be formalized as follows: *given a KB of axioms denoting the properties of the environment and of the agent's actions, a description of the initial state and of the goal, determine a* **plan** *that, when executed from the initial state, reaches a state where the goal is satisfied.*

Our notion of plan is given in terms of a transition graph [23, 14], which characterizes the dynamic behaviour of the agent for achieving a given goal. More specifically, the transition graph is labeled and nodes represent the knowledge state of the agents about the environment, while arcs represent actions to be performed in a given state.

The fundamental step towards the implementation has been to rely on the tight correspondence that exists between PDLs and Description Logics (DLs)

[32, 9]. By exploiting this correspondence, we have been able both to develop an interesting theoretical framework for reasoning about actions and to obtain an implementation that uses a knowledge representation system based on DLs.

The plan generation procedure implemented on the described logical framework is able to generate plans containing sequences of actions, if-then-else constructs associated to sensing actions, parallel execution of primitive actions (either ordinary or sensing actions), and simple forms of while loops (see [23, 14] for more details). The plan generated by our planer is partially correct in the sense that "if it terminates it leads to a state satisfying a goal" [14].

1.3 Coordinating Agents

Coordination in Multi-Agent Systems (MAS) is one of the most interesting areas of research in Artificial Intelligence and Robotics [11, 26, 24], since it allows for improving the effectiveness of a robotic system or cognitive agent both from the viewpoint of the performance in accomplishing certain tasks [11] and in the robustness and reliability of the system [26].

In order to coordinate a team of cognitive agents many design choices must be considered: (i) centralized vs. distributed system architecture; (ii) explicit vs. implicit communication; (iii) deliberative vs. reactive system. In the following a brief discussion is reported for each of these design choices, while a detailed description of the proposed solution adopted within our framework is described in the next sections.

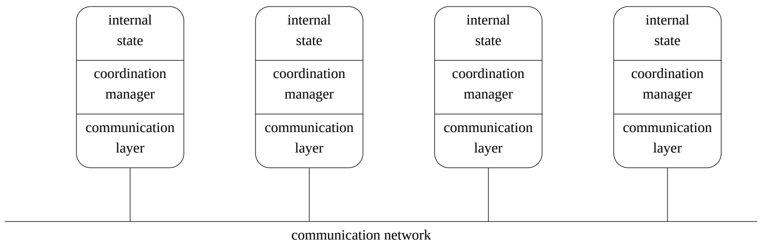


Fig. 3. System architecture for agent coordination

Centralized vs. Distributed. A centralized approach in the coordination of a Multi-Agent System is based on the presence of a special agent, that we call the *Coordinator*, that acquires and fuses information from the other agents in the system and, based on a global reconstruction of the state of the environment, is able to decide the actions to be performed by each agent in the system. A centralized approach is very effective and simple to implement, but it is strongly related to the reliability of the communication layer and to the ability in reconstructing global state of the world from local views of it. Moreover, if the Coordinator agent is not able for some reason to accomplish its task all the agents depending on the coordinator are not able to accomplish their tasks anymore.

On the other hand, in a distributed approach each agent decides the actions to perform on the basis of the information acquired both from the environment and, possibly, from the other agents. In this approach, the communication load is reduced and the robustness of the system is improved, but conflicts may arise among the agents, thus possibly decreasing the performance of the overall system.

In the realization of complex Multi-Agent System, it is also possible to integrate both a centralized and a distributed approach. For example, the centralized one may be used when a coordinator agent sends orders to a group of agents assigning them a specific task to be accomplished. However, within this group coordination of activities may be distributed for example as in [13], so that every single agent remains autonomous in its decisions, by still taking into account the global task assigned to it.

Explicit vs. implicit communication. Communication among agents in a Multi-Agent System is used in order to improve team performance, allowing the agents to acquire more information and to self-organize in a more reliable way. Communication can be both explicit in the sense that is based on a particular physical device (e. g. voice, or Radio transmitter) or can be implicit or stigmergic in the sense that the agents share informations among them changing the environment in a particular way (e. g. writing on the wall). Here we assume the use of explicit communication because if the device used for the communication is reasonably reliable (e.g. like a radio transmitter), direct communication is obviously more powerful than implicit one. An example of architecture for direct communication among agents is presented in Figure 3.

Deliberative vs. reactive system. In the context of Multi-Agent Systems by deliberative or reactive system we refer to the architecture of the overall system and not to the architecture of each agent. In this sense, with deliberative system architecture we denote a system architecture that allows the team to cope with the environmental changes by providing a strategy that can be adopted to reorganize the team members' tasks. Conversely, in reactive systems architectures every single agent in the team copes with the environmental changes by providing a specific solution to reorganize its own task, in order to fulfill the accomplishment of its originally assigned goal [20]. Our approach is based on a deliberative system architecture, in order to use all the resources available to the system to effectively achieve the global goal.

In addition to the above design choices several issues have to be considered, in order to address the problem of agent coordination, in particular we will discuss the following:

- Information Fusion
- Cooperative Sensing
- Task Distribution
- Action Synchronization

Information fusion In Multi-Agent System the information from the environment are extracted from different sources. In order to acquire a finer representation of the world those information need to be merged in a proper way. Several different approaches to the problem of information fusion can be found in the literature (see [17, 34]). For a specific analysis of multi agent approaches to information fusion see also [12].

Cooperative Sensing The issue of Cooperative Sensing is very similar to Information Fusion, but normally in Cooperative Sensing different agents coordinate in order to acquire a better world representation [28, 27], while Information Fusion is addressed in order to provide a better coordination.

Task Distribution When multiple agents need to coordinate in order to accomplish several tasks, an assignment of the task to the agents is needed. In particular, a task should be assigned to the agents that can provide the "best" solution for that task. The problem of choosing which agents is the best one to accomplish a given task is not trivial: in particular a very interesting problem of Task distribution is to choose the agents based on the current configuration of the overall system, thus providing a Dynamic Task Assignment [16].

Action Synchronization Often agents cooperating to accomplish a given task need to deal with resource sharing. In this case a synchronization for the actions of the agents is needed [1], in order to avoid interference between the coordinating agents and possible deadlocks.

In the present framework, we are specifically interested in *situation assessment*, that is the reconstruction of the emergency scenario as it evolves over time. Situation assessment involves both cooperative sensing and information fusion. Moreover, task distribution and synchronization are relevant to the distribution of all the tasks, that are involved in the rescue operations, including situation assessment.

2 ADK Functional Description

The Agent Developement Kit (ADK) is a framework that enables for the development of cognitive agents in an easy and structured way. Basically, the ADK defines a basic agent architecture and provides a set of tools and algorithms that are of general use in agent design and implementation. In this section, we present a functional model of the ADK. We start by defining its basic components, highlighting their relationships with the layered model previously described. Finally, we discuss in more detail the role and the functionalities provided by the ADK, without specializing them to the rescue domain, which will be addressed in the following.

ADK Agent Schema

In Figure 4 we describe the functional model of the Agent Development Kit. The planner is not shown in the figure, but it can be thought of as the module that generates plan library.

Before providing a description of the modules in Figure 4, we shortly introduce their functionalities:

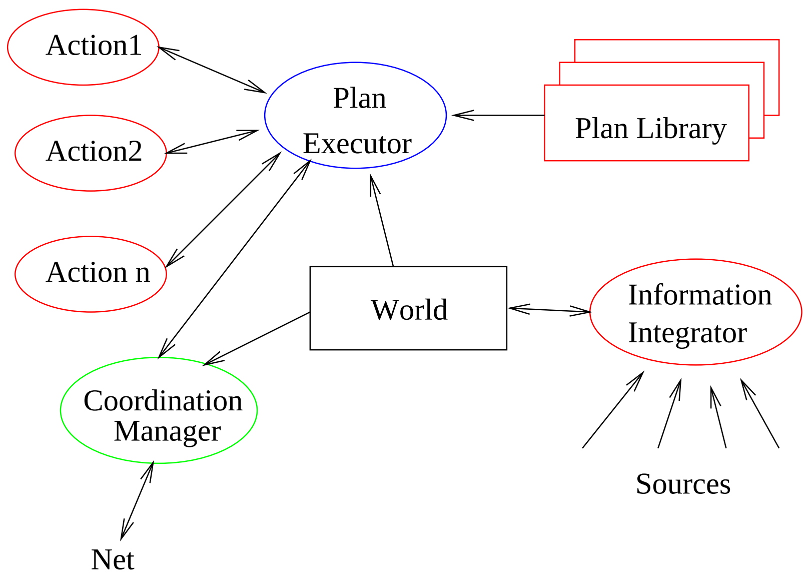


Fig. 4. Functional Agent Description

– *World*
is the basic data structure made up by world objects. It is dependent on the agent domain. The World data structure encapsulates three blocks of the *layered model*: the Numeric Model, the Symbolic Model and the Modeling block. The Numeric Model contains the feature level information about the world resulting from raw sensor data processing. The Symbolic Model contains the data expressed by the Numeric Model in a way suitable for reasoning, and the Modeling block performs the translation from Numeric to Symbolic.
Currently in the ADK world only the feature level is stored, while an on demand evaluation of the Symbolic Model is performed when requested by the Plan Executor.
– *Plan Library*
is a data structure that contains a set of off-line generated plans. It is dependent on both operating domain and agent goals. The plan library contains a plan for each of the goals the agent can pursue. The plan library toghether with the plan executor can be related to the subset of the *Reasoning System* of the layered model, which takes care of scheduling a given plan. Although the layered model of [21] allows for on-line planning, the ADK agent is based on an off-line planning whose goal is to generate a plan library that feeds the scheduler.
– *Plan Executor*
is the module that chooses a plan from the Plan Library in order to reach the specific goal set by the coordination handler. It performs the plan execution

by starting/stopping one or more domain dependent Actions, on the basis of some condition obtained by World analysis.

- *Information Integrator*
  is the module that refreshes the World based on the information coming from external sources and information generated by the processing of onboard sensor inputs.
- *Coordination Manager*
  is the module that, analyzing the current World state and the other agents coordination information, chooses the agent specific goal in order to accomplish the global task. It also sends the coordination information to other agents.
- *Actions*
  the basic actions are the atomic operations through which the agent can modify the world state. The ADK model basic actions set corresponds to the *Control System* of the layered model.

Notice that the layered model presented in Figure 2, that is designed for single-agent systems, is not fully reflected in Figure 4, which deals with a multi agent framework. In particular, here we simply refer to the plan execution component at the deliberative level, while the operational level is not included for simplicity. Nonetheless, some of the Information Fusion tasks that are handled by the Information Integrator module may be accomplished with numeric representations that are used in the operative level of the architecture. Consequently, although this is not explicitely represented the layered architecture Figure 2, is actually assumed here.

Below, we provide a more detailed description of the Plan Executor, Information Integrator and Coordination Manager modules, because they are of general interest in agent design. We leave apart the action description because they are strongly related to the particular application domain.

2.1 Plan Executor

As mentioned in 1.2 a plan is defined as a transition graph that, if successfully executed, allows our agent to reach the goal. Below we describe first the representation of plans, then the plan execution module, while its implementation is addressed in the next section.

A remark concerning the overall approach to planning is in order. In the definition of our system architecture we have chosen to generate a library of plans to be used in typical off-line situations. Therefore, during the mission, the agent has a library of plans and can select the one that is adequate for the situation at hand. Notice that off-line generation in our framework is not a severe limitation, since the use of sensing actions allows for deriving general plans that are specialized during their executions according to the actual sensing of some properties. In principle, it is possible to generate universal plans that guarantee to reach a goal whatever is the initial state, by sensing all that is needed to know during plan execution. Here we adopt an intermediate solution, where specific plans are designed off-line and may be specific to some initial conditions. Plans, however may include sensing actions, whose result is determined only at run time by acquiring information on the situation at hand.

Plan Representation A plan is represented as a *transition graph*, where each node denotes a state, and is labeled with the properties that characterize the state, and each arc denotes a state transition and is labeled with the action that causes the transition. A state represents a situation the system can be in and is characterized by a set of properties which give a (complete) description of the situation. However, the plan may include only a subset of the properties that are needed in order to accomplish the transitions in the graph.

Actions are represented using preconditions and effects. Preconditions are the conditions that are necessary for activating the action and indicate what must be true before the action is executed: they specify circumstances under which it is possible to execute an action. Effects are the conditions that must hold after the execution of the action and characterize how the state changes after the execution of the action: they specify direct effects of an action if executed under given circumstances. A description indicating the overall behaviour of the action is associated with each action, and is only concerned with the execution of the action in that particular context.

The actions in the graph can be classified into ordinary (i.e. movement) actions and sensing actions. The former cause changes in the environment, while the latter permit the acquisition of information, in order to let the robot take better decisions. Both actions are relevant to state transition. Sensing actions are goal-directed behaviours used to verify the value of a property in the world. Sensing actions are therefore associated with conditions to be verified: depending upon the condition which is true, a different part of the plan is executed. The plan can also include parallel execution of primitive actions when their preconditions and effects do not conflict, i.e. do not generate inconsistencies.

In the formal framework that we have adopted, a number of assumptions are made both to limit the complexity of the language and to enable for automatic reasoning. Consequently, for an effective execution of plans, additional information concerning the verification of preconditions and effects is needed. In particular, both the duration of actions and the failures of action execution must be taken into account.

Both preconditions and effects can be interpreted in different ways by the plan executor. Some preconditions must be constantly verified during the entire execution of the action, while others need to be checked only for the action activation. In the first case, the action is carried out as long as the condition is true. If the condition becomes false during the execution of the action, an exception occurs and the action fails. This means that there is no state transition depending on that action execution.

Similarly, some effects determine the action termination and the state transition, while others are side effects of the action but do not cause the state transition.

Following the considerations above, the plan must be marked with additional information, that is not in the plan generated according to the specification, but necessary to the monitor for interpreting and executing the plan. In particular, the monitor has to know which preconditions have to be verified during the entire execution of the action and which effects determine the state transition.

Based on the above observation, we can now introduce an *Executable Plan*. Given a set of ordinary actions $A = \{a_i\}$, and a set of sensing actions $C = \{c_i\}$, a plan is an ordered graph $G =< V, E >$ where $V = \{s_{init}, s_{goal}, s_1, \dots, s_n\}$ set of vertex made by plan states, and $E = \{< s_i, s_j >\}$ set of edges meaning the action set that can perform the transition from the state s_i to the state s_j.

For convenience, in the case of concurrent (primitive) actions, each edge is labeled with a set of actions, which represents the parallel execution of the actions. Moreover, to denote the conditions that may cause the termination of actions, or the failure, we introduce a new type of edge, called *trigger*. Trigger edges correspond to terminating conditions for an action, or action failure. In the first case, the subsequent state is the state resulting from the effects of the action and corresponds to the destination state of the action edge. In the second case, a failure state may not be explicitly represented if the plan can successfully continue from some other state already present in the plan. When it is not possible to resume the execution from a state within the plan, the plan execution fails and a new plan must be selected.

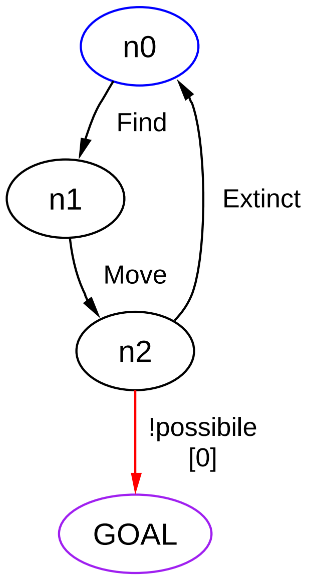


Fig. 5. Example of Executable Plan

Plan Execution The Plan Executor is in charge of the correct execution of the actions composing the plans. Its task is that of visiting the graph, calling and suspending the actions as necessary.

More specifically, the Plan Executor takes as input a set of plans contained in the Plan Library, possibly some initial conditions that are verified in the current situation, and a goal, which is provided by the Coordination Manager. The Plan Executor selects the plan for achieving the goal for reaching the given goal.

Then the plan is executed, by initially activating the actions originating in the initial state and then by performing the following steps:

1. Check action termination. If action terminates, a new state is reached and the action starting in the new state are activated. Termination is verified by taking into account also the trigger edges.
2. If a failure state is reached, the plan is aborted and a new plan must be selected.
3. It the goal is changed then the plan is aborted and a new plan must be selected.

2.2 Information Integrator

It is very often the case that an agent is equipped with a large variety of sensors and/or sensing capabilities, each one providing different information. The task of the *sensor fusion* process is to calculate the new state of the world starting from the previous states and the current sensor information. Moreover, in a typical scenario we consider a set of communicating and coordinating agents. Each of them has its local sensors and receives (through communication) the local information percepted by the others; in this case, the information communicated by the other agents have to be taken into account in reconstructing the new state, because other agents may send information about world parts that are hidden to the agent or that can be used to correct or refine agent's sensing errors. Once information about the world state is received, an agent can choose to update its internal state, or to discard part/all of the incoming information depending upon the situation, and to the estimated reliability of the sender. Summarizing, the Information Integrator process should do both sensor fusion and high level information fusion.

To solve the information fusion problem, a lot of tecniques and algorithms have been proposed, depending on the specific domain of the fusion and on the abstraction level of the information being integrated. At the lower level, we find the fusion algorithms that work using the onboard sensors as input, extracting features or even more complex information from a set of sensor readings. Problems as localization or obstacle detection belong to this category. At a higher level, we find the processes that take care of integrating high level information coming from external sources, in order to achieve a wider environment knowledge, that is called high level fusion or situation assessment *information fusion* [18]. Although it is possible to integrate in our agent structure also a sensor fusion capability, in the following we mainly deal with high-level information fusion.

Several issues could arise at this level, for instance some of the transmitting agents may be not operational, or may send incorrect information. The information integrator has to detect such situations, and make use of all the received information to reconstruct a correct view of the environment and thus improve the decision making process.

We propose the information integration model described in Figure 6.

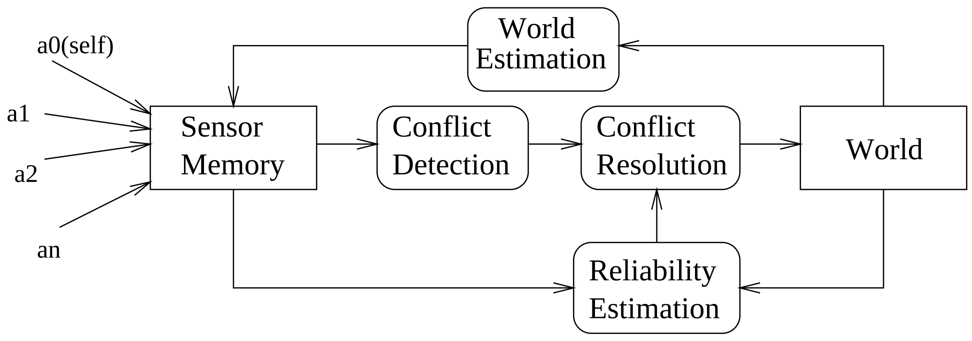


Fig. 6. Sensor Integration Schema

At each time step the Information Reports coming from the different sources are collected in the Sensor Memory. There are three main sources of information that can write in the sensor memory:

– the on board sensing processes
– other agents, via network
– the world estimation module

In the Sensor Memory all the reports are collected, without considering possible conflicts. At each step a Conflict Detector Module analyzes the Sensor Memory and finds out the possible conflicts. Then the Conflict Resolver tries to assess the current situation, by using the Reliability Estimator, that takes care of evaluating the reliability of the information sources, and the current world estimation. The World Estimator, given the domain dependent evolution laws, projects the current state into a suitable future one, that will also be considered by Conflict Resolver at the next step.

2.3 Coordination Manager

The Coordination Manager has the fundamental goal of allowing the agents to share information about their intentions in such a way to avoid possible conflicts in the execution of the tasks and to cooperate in the achievement of global goals.

In our framework, we have distinguished two kinds of agents: a first group (called *operative agents*) that are responsible for the actual operation in the environment, a second one (called *coordinator agents*) that collect data from all the agents, perform the process of information fusion and are able to take decisions about the action to be performed by the other agents. Moreover, the coordination agents may interact among each other in order to define a global strategy for solving a given problem.

There are thus two kinds of *coordination messages* that are exchanged by the agents: 1) **requests**, that are sent from a coordinator agent to a group of operative agents assigning them a goal to be achieved; 2) **intentions**, that are exchanged among the members of a group of agents working for the same global goal or among the coordinator agents in order to coordinate their activities for accomplishing the assigned task.

Therefore in a general complex application of Multi-Agent System, it is possible to devise a scenario in which there are several groups of agents with different operative capabilities in the environment that are coordinated by a coordination agent and a set of virtual communication networks allowing for message exchanging among these agents.

An example of such a network topology is given in Figure 7, which shows two groups of operative agents, with the corresponding communication networks, and two coordinator agents that can communicate with their operative agents as well as among each other.

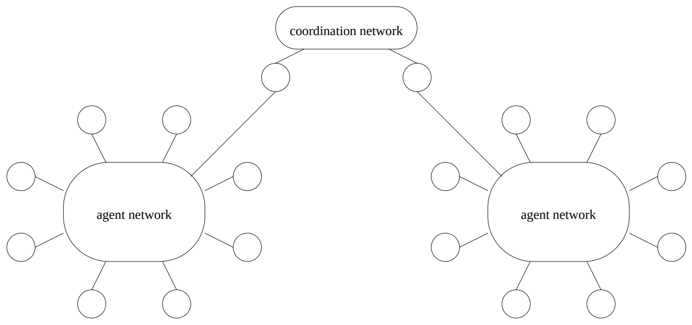


Fig. 7. Communication architecture for a complex MAS

3 ADK Implementation

In this section we discuss in more detail the behaviour of some modules provided by the ADK, with the goal of providing some detail about the implementation and the actual definition of agents that are not dependent on the operating domain. Then, we present a general procedure for designing agents.

3.1 World

World is the data structure that contains the knowledge about the external world, owned by an agent at a given instant of time. World allows for an easy retrieval of the contained information. Although the World description is structured according to the specific domain, here we consider it as a World Object container. Each World Object instance models a specific Object belonging to the real world and resulting from the information integration process.

A World Object is the base item contained in the world. It is constituted by a set of Properties, each of them uniquely identified in the containing World Object by a Property Key. A Property can be an Object, allowing for a hierarchical world representation. For each Property we want to know the last source(s) that updates its value and the time the property was updated.

As an example of World Object example, consider a building that has a set of attributes like the number of floors, the first floor plant and so on...; each simple attribute (like the number of floors) is modeled by a property, the complex ones (like the plant) are modeled by objects.

World objects should be exchanged between agents. Therefore, they must exhibit the following properties: introspection, serializability and uniqueness.

The serialization mechanism allows different agents to uniformly represent their world knowledge. Self-description (introspection) allows a remote agent to interpret any object structure. Uniqueness is defined with respect to the container: an object has to be uniquely addressed through its *Object Identifier* in the container object, within all the same-level objects. Therefore, the path from the object root uniquely identifies each object component.

Figure 8 presents a tree representation of a complex object. Observe that object identifiers are unique within their container, that is the parent object in the tree view.

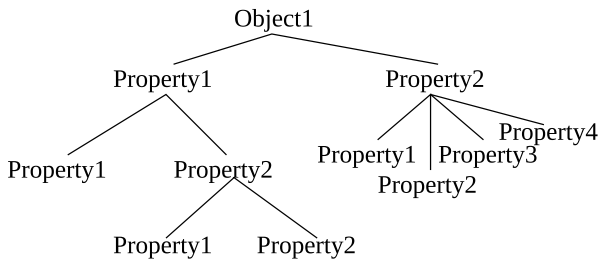


Fig. 8. Instance of a complex World Object

3.2 Plan generator

The representation of plans is done by exploiting an objetc-oriented approach. This has several advantages from the implementation standpoind, being modular, extensible, readable and reusable. In particular, by defining each plan as an object of the *pemPlan* type, which is also a class derived from *pemAction*, a plan can be viewed as a primitive action at a higher level of abstraction. This definition allows plans and actions to be treated in the same manner, combining them in plans of higher hierarchical order [33]. In a higher level plan, external conditions determine the next plan to be activated. This feature also allows for a great flexibility in the design of plans, although from a theoretical standpoint the "safe" parallel execution of parallel plan would require a deeper analysis of the intermediate effects caused by the execution of each plan, with respect to the execution of the other plan. However, from a practical pont of view, such interactions are handled by the execution mechanism, which can identify the possible plan failure.

The plan can be designed through the help of a graphical tool that is described in Section [7].

The Plan Executor is essentially made by two components which correspond to the plan selection and to the stepwise execution as described in Section 2.1. The only technical issue arises from the management of the hierarchical structure of plans that must be correctly handled by the execution in order to guarantee the proper termination/activation of actions.

3.3 Information Integrator

The Information Integrator modules that, given the description of the World in Section 3.1 in term of Objects and Properties, are not dependent on the context are the Sensor Memory and the Conflict Detector, that are discussed in the following. A simple implementation of the Conflict Resolver that uses an algorithm based only on the general object features is also given.

Sensor Memory The basic input for the information fusion process is the *report*. A report is an atomic observation of the environment. If the subject (the one who performs the observation), the object (the world fragment interested by the observation) of the observation, and the time at which the observation occurs are known, then we can uniquely address the value of such an observation. Since we can uniquely identify an atomic property [1] of a world object, simply by chaining the ObjectIds from the root to the leaf of the tree world representation, then we can uniquely identify a report by a key in the form $\langle ObjectId^*, Justification, Time\rangle$; where $ObjectId^*$ means the Object Id chain from the root to the leaf; $Justification$ means the source of the observation and $Time$ means the source of such an information.

[1] An atomic property is a property that does not contain inner properties. Referring to the tree representation of an object, the atomic properties are the leaves.

Since some stages of our fusion processes require the use of the reports at the previous times, then we collect them in a structure: the Sensor Memory. The Sensor Memory is an associative container that allows to fetch a report given its key.

Conflict Detector We define a conflict as a situation in which at the same time the same property is asserted by different sources with different values; formally, a conflict on property p at time t holds if $\exists\ p, t, j_1, j_2\ \bigwedge\ o(p, t, j_1) \neq o(p, t, j_2)$ where j_1 and j_2 can be two different sources and $o(\cdot)$ is the property value. Thus we can extract the conflicts that arise in a particular sensor memory configuration simply by looking at each set of reports belonging to the same time and referring to the same property. By partitioning the above set in classes that have the same value, we classify the report sources on the basis of the property value they asserted.

The Conflict Detector is the module that performs such an operation. It analyzes the Sensor Memory and extracts a set of ordered reports, in the form of

$$\{\langle p, t\rangle \rightarrow \{\langle v, J\rangle\}\}$$

where v denotes a property value and J denotes the set of sources that at time t asserted the property p to have the value v. A conflict arises on $\langle p, t\rangle$ if $|\{\langle v, J\rangle\}| > 1$.

Conflict Resolver The Conflict Resolver is the module that takes care of updating the World in a consistent way, actualy, it adopts a particular policy to solve the conflicts detected in the previous stage. In doing that, a lookup in the Sensor Memory and an estimation of the Sources Reliability may be needed. More complex policies can be easily defined, see [7].

Currently, only one simple policy is implemented: if a conflict arises, the winner value is the one detected by the agent itself through onboard devices. If such a value does not exist, the property value belonging to the greater justification set J is chosen.

3.4 Coordination manager

The implementation of the coordination manager is divided in two modules: the first is a communication layer that allows the agents to exchange messages among them, the second is the dynamic task assignment performed by a group of agents sharing the same goal.

The communication facilities needed for these two modules are based on broadcast communication and on a publish/subscribe mechanism such that every agent receives only those messages that are useful for accomplishing its task (i.e. coordination messages and messages from the other members of its group).

The coordination protocol implemented for dynamic task assignment is based on the computation of *utility functions*, that denote the ability of every agent to perform a task and that are computed by every agent periodically during the mission, based on its current knowledge about the world. The values of the utility functions are exchanged among the agents in order to assign specific tasks to the agents that are in the better condition to accomplish them. The details of this distributed coordination protocol have been described also in [5].

3.5 Designing an agent

The agent design is structured according to the following steps:

1. plan design
   (a) definition of plan structure
   (b) ordinary actions definition
   (c) sensing actions definition
2. information fusion definition
   (a) conflict resolution policy definition
   (b) reliability estimation policy definition
   (c) transmission policy definition
   (d) update reception policy definition
3. coordination handler definition

Plan Design The first stage in the agent design is to characterize its behaviour by defining a plan. In this context, we focus on plan design by means of a graphical tool. However, through the formal approach discussed in Section 1.2, the plan could be automatically generated from an axiomatization of the actions that can be accomplished by an agent (see [23, 14]).

Plan design is supported by a tool called *Plan Assistant*, that is part of the ADK. The Plan Assistant provides a GUI for drawing plan graphs; moreover, it performs some executability check before saving the drawn plans. It is possible to label each edge with an action, that can be either a primitive action or the name of a plan previously defined.

As already mentioned, the edges are used to represent both sets of actions, to be executed concurrently, and triggers, that are used to handle action termination and switch to the next state.

An additional distinction is made among the ordinary actions that are executed in parallel. Some of them are considered *auxiliary*, since their termination does not influence the termination of the set of actions concurrently executed.

After defining the structure of the plan(s), primitive actions must be defined. Primitive actions can be of two kinds: ordinary actions and sensing actions. A primitive action is a process that can be:

– initialized
– terminated
– checked for its execution status
– executed

Each of such functionalities has to be handled by proper code. A difference from a sensing action and an ordinary one is that the execution of a sensing action is expected to return a value about the world property being checked.

In addition, for each action preconditions and effects should be specified, together with the execution annotations (as explained in 2.1), although they are currently handled by the code associated with the primitive actions.

Information Fusion Strategy As previously said ADK implements an information fusion module. Here we sketch the steps for developing an information fusion module, operating in a general scenario.

In order to define a fusion strategy the following tasks have to be faced:

- To define a conflict resolution policy, that means to define how to handle the case of two or more groups of sources asserting at the same time that a property of the same object has different values.
- To define what kind of information a source has to transmit. The main problem in this stage is the trade-off between accuracy in reconstruction and communication load, accordingly with the system specifications.
- To define a strategy for assigning a reliability value to the sources, on the basis of the owned knowledge.
- To define a World Estimation module. Such a module has the role of forward propagate the agent knowledge on the basis of the current one, avoiding/filtering wrong updates in the state.

Coordination The definition of coordination capabilities is based on the definition of the communication capabilities of the agent, which are specified by defining the type of the agent.

In the ADK, the coordination module has the task to determine the agent specific goal. When a centralized approach to coordination is chosen, the coordinator agent must be equipped with the procedure for task assignment and resource allocation, while the operative agent simply executes the goal as received by the coordinator agent.

When a distributed approach to coordination is chosen, the agent must determine the goal based on:

- an estimation of the world
- the intentions of the other robots

Such information are communicated through the network according to the group structure.

Specifically, the method we adopt for distributed coordination requires the definition of a utility funcion for each task that can be performed by an agent (see [5]). Such a function provides an estimation of how effectively the agent can accomplish the task.

4 Conclusion

In this document we have addressed the design of systems of Cognitive agents, characterized by being situated in a partially known, dynamic, unpredicatable environment. In particular, under the assumption that agent can communicate, we have considered the problem of situation assessment, which involves both information fusion and cooperation among the agents.

In addition, we have described the functionalities of a tool to support the design of agents with the above mentioned capabilities, called Agent Development Kit. This tool allows the multi agent system designer to specify the behavior of an agent by defining the plans that allow the agents for the execution of its task, its information fusion strategies, as well as its capability to cooperate with other agents.

The ADK has been fully implemented, and its practical application in the rescue domain is in progress (under the "Project Real-time planning and monitoring for search and rescue operations in large-scale disasters" funded by Italian National Research Council). Some preliminary results are also reported in [7].

References

1. R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi robot cooperation in the martha project. *IEEE Robotics and Automation Magazine (Accepted in the Special Issue on "Robotics and Automation in Europe : Projects funded by the Commission of the European Union*, 1997.
2. Michael Bowling. Robocup rescue: Agent developement kit. *http://robomec.cs.kobe-u.ac.jp/robocup-rescue*, 2000.
3. Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1), 1986.
4. C. Castelpietra, A. Guidotti, L. Iocchi, D. Nardi, and R. Rosati. Design and implementation of cognitive soccer robots. In *Proc. of RoboCup Symposium*, 2001.
5. C. Castelpietra, L. Iocchi, D. Nardi, M. Piaggio, A. Scalzo, and A. Sgorbissa. Coordination among heterogenous robotic soccer players. In *Proc. of International Conference on Intelligent Robots and Systems (IROS'2000)*, 2000.
6. Robocup Rescue Technical Committe. Robocup rescue simulator manual v0.4. *http://robomec.cs.kobe-u.ac.jp/robocup-rescue*, 2000.
7. F. D'Agostino, A. Farinelli, G. Grisetti, L. Iocchi, and D. Nardi. Adaptability of the rescue simulator. *Technical Report*, 2002.
8. Giuseppe De Giacomo, Luca Iocchi, Daniele Nardi, and Riccardo Rosati. A theory and implementation of cognitive mobile robots. *Journal of Logic and Computation*, 5(9):759–785, 1999.
9. Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI'94)*, pages 205–212, 1994.
10. Mattia De Rosa, Luca Iocchi, and Daniele Nardi. Knowledge representation techniques for information extraction on the Web. In *Proceedings of Webnet 98*, 1998.
11. D. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, 1996.

12. A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, and Salerno M. Information fusion. *Technical Report*, 2002.
13. A. Farinelli, L. Iocchi, G. Grisetti, and D. Nardi. Coordination in dynamic environments with constraints on resources. In *IROS Workshop 2002 on Cooperative Robotics*, Lausanne, Switzerland, October 2002.
14. A. Farinelli, L. Iocchi, G. Grisetti, and R. Nardi, D.and Rosati. Generation and execution of partially correct plans in dynamic environments. In *AAAI-02 Workshop on cognitive robotics*, 2002.
15. Erann Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, 1992.
16. Brian P. Gerkey and Maja J Matarić. Principled communication for dynamic multi-robot task allocation. In *Proceedings of the International Symposium on Experimental Robotics*, Waikiki, Hawaii, December 2000.
17. D. L. Hall and J. Llinas. *Handbook of multisensor data fusion.* CRC Press, 2001.
18. D.L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, pages 6 – 23, 1997.
19. Steve Hanks and R. James Firby. Issues and architectures for planning and execution. In *Proc. of Workshop on Innovative Approaches to Planning, Scheduling, and Control*, 1990.
20. L. Iocchi, D. Nardi, and M. Salerno. Reactivity and deliberation: a survey on multi-robot systems. In E. Pagello M. Hannebauer, J. Wendler, editor, *Balancing Reactivity and Deliberation in Multi-Agent Systems (LNAI 2103)*, pages 9–32. Springer, 2001.
21. Luca Iocchi. *Design and Development of Cognitive Robots.* PhD thesis, Univ. "La Sapienza", Roma, Italy, On-line `ftp.dis.uniroma1.it/pub/iocchi/`, 1999.
22. Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Planning with sensing, concurrency, and exogenous events: logical framework and implementation. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, 2000.
23. Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Planning with sensing, concurrency, and exogenous events: logical framework and implementation. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 678–689, 2000.
24. H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. Robocup: A challenge problem for AI and robotics. In *Lecture Note in Artificial Intelligence*, volume 1395, pages 1–19, 1998.
25. Y. Lesperance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter, and R.B. Scherl. A logical approach to high-level robot programming. In *AAAI FAll Symposium on Control of the Physical World by Intelligent Systems*, 1994.
26. Lynne E. Parker. ALLIANCE: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.
27. Lynne E. Parker. Cooperative robotics for multi-target observation. *Intelligent Automation and Soft Computing*, 5(19), 1999.
28. I. Rekleitis, G. Dudek, and E. Milios. Multi-robot collaboration for robust exploration. In *Proceedings of International Conference in Robotics and Automation*, 2000.
29. S. Rosenschein. Plan synthesis: a logical perspective. In *Proc. of the 8th Int. Joint Conf. on Artificial Intelligence*, 1981.
30. S.J. Russel and P. Norvig. *Artificial Intelligence, a Modern Approach.* Prentice Hall, 1998.

31. A. Saffiotti, K. Konolige, and E. Ruspini. A multivalued logic approach to integrating planning and control. *Artificial Intelligence*, 76:481–526, 1995.
32. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence*, 1991.
33. Reid Simmons. An architecture for coordinating planning, sensing and action. *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, 1992.
34. L. Valet, G. Mauris, and P. Bolon. A statistical overview of recent literature in information fusion. *IEEE Aerospace and Electronics Systems Magazine Volume 16 issue 3*, pages 7 – 14, 2001.

Monitoring and Information Fusion for Search and Rescue Operations in Large-scale Disasters

Fabrizio d'Agostino
fdagosti@dis.uniroma1.it

Alessandro Farinelli
farinelli@dis.uniroma1.it

Giorgio Grisetti
grisetti@dis.uniroma1.it

Luca Iocchi
iocchi@dis.uniroma1.it

Daniele Nardi
nardi@dis.uniroma1.it

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 ROMA, Italy

Abstract - *The goal of the project, which is currently under development, is to design tools to monitor the situation after a large-scale disaster, with a particular focus on the task on situation assessment and high-level information fusion, as well as on the issues that arise in coordinating the agent actions based on the acquired information. The development environment is based on the RoboCup-Rescue simulator: a simulation environment used for the RoboCup-Rescue competition, allowing for the design of both agents operating in the scenario and simulators for modeling various aspects of the situation including the graphical interface to monitor the disaster site. Our project is focussed on three aspects: modeling in the simulator a scenario devised from the analysis of a real case study; an extension of the simulator enabling for the experimentation of various communication and information fusion schemes; a framework for developing agents that are capable of constructing a global view of the situation and of distributing specific information to other agents in order to drive their actions.*

Keywords: disaster simulation, multi-agent systems, information acquisition, situation assessment, information fusion, planning.

1 Introduction

Search and rescue of victims in large-scale disasters are not only highly relevant social problems, but pose several challenges from a scientific standpoint. When earthquakes, eruptions or floods happen, a considerable organizational capability to aid the disaster victims as fast as possible is required. However, too often different secondary disasters, connected with the main one, occur, which avoid the correct execution of a rescue plan a priori decided. For example, as reported for the Kobe earthquake (1997) [12], after the earthquake, fires arose between the debris of the destroyed wooden houses, communication infrastructures and transportation systems were largely damaged, causing additional difficulties for the aids.

The goal of the present project is to develop software tools to support the management of this kind of emergency, more specifically to design a support system for search and rescue operations in large-scale disasters, both for prevision and training as well as for operative actions. Even though there are significant results on the design of robots to support search and rescue [14], here we are specifically concerned with the design of a software tool to support both the situation assessment and the planning/control of operations; moreover, we are interested in the deployment of the techniques for achieving cooperation in a multi-agent system [18].

In this context, the RoboCup initiative, the organization which, starting from 1997, arranged the world championship for soccer player robots [9], proposed a new scientific challenge named RoboCup-Rescue [10, 11, 19], where the problems faced are those of bringing aids in a large disaster. The RoboCup objective is to create system based on AI and Robotics technologies, where heterogeneous agents (software, robots, human beings) interact in a cooperative manner. In particular, we are concerned with the possibility of deploying and extending the RoboCup-Rescue Simulator [17], a simulation system whose main feature is the ability to deal simultaneously with many events, thus allowing for the study of different rescue strategies with agents, autonomous or not, and to facilitate the development decision support systems working in real-time.

There are several technical issues that arise in order to pursue the design of a tool like the RoboCup-Rescue simulator: event modeling and simulation, integration and visualization of data, resource management, planning and scheduling, execution monitoring. We are specifically interested in the problems arising in the interaction of a large number of heterogeneous agents [18], i.e. the members of a rescue team. Another relevant research aspect, strictly related to the previous one, is concerned with the coordination of the agents themselves, while they are trying to achieve a common goal [3, 6]. In particular, we address the problem of situation assessment (information fusion) in the disaster scenario, within a multi-agent system.

In order to ground our project in a real scenario we have chosen to access the data of the Umbria and Marche earthquake (1997) and to consider the structures and strategies currently adopted by the Italian VVF (Fire-Dept) [5]. This, not only provides us with a significant body of expertise on rescue operations, but also gives us the opportunity to set up a prototype experimental setting to show the results of the project. The specific elements of the domain under consideration, as well as the relevant features of the RoboCup-Rescue simulator are described in Section 2.

In a complex domain, such as the one of search and rescue operation the agents must show both perception and fusion capabilities as far as the acquisition of information about the operation scenario is concerned; in addition, agents should be able to act reactively as well as to plan and execute complex actions, dealing with failures and continuous changes in the environment. In order to identify the functionalities to be provided to the agents for situation assessment, we have done an extensive survey of the literature on information fusion, specifically looking at the application of multi-agent approaches. The outcomes of this work are summarized in Section 3.

In order to provide the above mentioned capabilities the agent must rely on hybrid achitecture that is both heterogeneous (dealing with different representations of information) and asyncronous (to effectively integrate reactivity and planning) [8]. In Section 4 we describe an agent model and the Agent Development Kit (ADK) that we are developing to support the modeling and implementation of agents embodying the features required for the search and rescue simulation.

We conclude the paper by making some considerations on the exploitation of the proposed approach within Italian VVF and on the research problems that are currently under investigation.

2 The RoboCup-Rescue Simulator: an application to the earthquake of Umbria and Marche

The RoboCup-Rescue Project started in 1999 with the goal of developing a comprehensive urban disaster simulator (see [4]). It aims at producing a software environment useful for both testing intervention strategies in a virtual world and supporting decisions in case of real disasters such as earthquakes or big fires. Below we sketch the overall structure of the simulator to provide some indications on the components that need to be developed in order to apply the simulator to a specific disaster scenario. For a detailed description of the simulator see [17].

The RoboCup-Rescue Simulator has a distributed architecture, formed by several modules, each of them being a separate process running in a workstation on a network. The following are the main components of the simulator:

- *Geographic Information System* - The GIS module holds the state of the simulated world. Before simulation begins, it is initialized by the user in order to reflect the state of the simulated area at a given time, then it is automatically updated at each simulation cycle by the kernel module.

- *Kernel* - This module is connected to any other module. At each step it collects the action requests of the agents and the output of the simulators, merging them in a consistent way. Then, the kernel updates the static objects in the GIS and sends the world update to all the connected modules.

- *Simulators* - Fire-simulator, Collapse-simulator, Traffic-simulator, etc. are modules connected to the Kernel, each one simulating a particular disaster feature (fire, collapses, traffic, etc.). At the beginning of every simulation cycle, they receive from the kernel the state of the world, then they send back to the kernel the pool of GIS objects modified by the simulated feature (for example, a pool of burned or collapsed buildings, obstructed roads, etc.)

- *Agents* - Agent modules are connected to the kernel and represent “intelligent” entities in the real world, such as civilians, police agents, fire agents, etc. They can do some basic actions, such as extinguishing a fire, freeing obstructions from roads, talking with other agents, etc. Agents can also represent non-human entities: for example they can simulate a police-office, a fire station, an ambulance-center, etc.

- *Viewers* - Their task is to get the state of the world, communicating with the Kernel module, and graphically displaying it, allowing the user to easily follow the simulation progress.

In order to use the RoboCup-Rescue simulator in the context of the present project several issues must be taken into account. The first issue we have addressed is the choice of the domain that should be based both on the availability of data and on suitability of the RoboCup-Rescue simulators in modeling such an area. It is beyond the scope of the present project to design specific disaster simulators. After the domain has been identified the representation of the information concerning the disaster scenario to be used in the simulation must be constructed. Then, the features of the agents need to be analyzed to verify whether they are suitable for the modeling of the scenario. While the domain and its representation are described in the rest of this section, subsequentely we consider architectures and agent systems for information fusion, before addressing the design of agents in the search and rescue scenario.

2.1 The earthquake of Umbria and Marche

Throughout the fall of 1997, a serious earthquake affected the italian regions of Marche and Umbria: many housing estates as well as important artistic and religious monuments were heavily damaged, first and foremost the world-famous Basilica of S. Francesco in Assisi. In order to experiment and verify techniques and methodologies developed in our work, we have selected Foligno, one of the most important cities in that region, as an interesting scenario for running a disaster simulation; below we discuss the features of the chosen site and describe the main aspects of its representation within the simulator. In order to build such a representation we have developed a graphical editor that, starting from a bitmap of the site, supports the input of the description.

2.2 Domain features

Foligno is located in a flat region of eastern Umbria. Its urban structure is characterized by a medieval center surrounded by more recent suburbs; in particular we are fixing our attention on an area of about 1 km^2 in the city center.

In the area under consideration there are no high-rise buildings; most recent structures are mid-rise, in the four- to nine-story range. All large, multi-story buildings were constructed of reinforced concrete. Oldest buildings were mainly constructed of rubble-work, whereas only few structures are steel frame buildings or wood buildings. There are no industrial structures; most buildings are housing estates having variously-shaped plants. The road network is quite irregular, with not very large roads and narrow alleys. The following paragraph describes the model we adopted to represent this domain in the simulated world. The following paragraph describes the model we adopted to represent this domain in the simulated world.

2.3 World model

The world model we adopted is derived from the RoboCup-Rescue simulator model; it is somewhat minimal, but it could be easily extended to fit real scenarios more closely. It deals with three main entities or *object classes*: *buildings*, *roads* and *nodes*, respectively. The road network is described by a graph having one or more edges for each road and one node for each crossroad and for each junction between adjacent edges constituting a road. Also, a node can represent a linkage point (*access-point*) between a building and a road. Each object class (*building*, *road*, *node*) is characterized by a number of attributes describing a specific instance of the class. The following paragraphs show the features of each class.

2.3.1 Buildings

Building objects represent every kind of building on the map: houses, police offices, hospitals, fire stations, ambulance centers, refugees, etc. As a result of a earthquake shock, a building can collapse and obstruct a road; moreover, a building can catch fire more or less likely, according to its constituent material; for example, a concrete building is less flammable than a wooden one. Further, buildings can have one or more floors and one or more linkage points with the surrounding roads. The main attributes of buildings are: *Plant*, *Kind*, *Material*, *Fieryness*, *Brokenness*, *Floors*, *Entrances*.

2.3.2 Roads

Road objects are the edges of the road network graph; they represent every street, lane, tunnel, bridge, etc. in the map. A road can be partially or totally obstructed by rubble in consequence of the collapsing of an adjacent building. Further, a road has one or more traffic lanes on each side and can have a sidewalk or not. The most relevant road attributes: *Kind*, *Length*, *Width*, *Block*, *Repair-Cost*, *Lines-to-head*/*Lines-to-tail*, *Sidewalk-width.*

2.3.3 Nodes

Nodes represent crossroads or linkage points between buildings and roads. Moreover, a whole road can be split into two or more adjacent edges, connected to other nodes. The following are the most important attributes of this class: *Roads, Signal, Signal-timing*.

3 Architectures and Agents for Information Fusion

Information fusion is broadly used in various application fields such as defense, geoscience, robotics, health, industry and many techniques have been developed for the fusion process (see for example [7, 20]). Moreover, the degree of abstraction of information (raw data, features, or symbols) that is used in the fusion process is an important design element to take into consideration in the development of an information fusion system. In this work we are mainly interested in the definition of the system architecture that is required for implementing a system of robotic agents acting in a rescue domain like the one described in the previous section.

The design and development of system architectures is a central issue in the design of a fusion system. All the architectures proposed in the literature may be grouped in three categories:

- Centralized
- Hierarchical
- Distributed

The centralized approach to the development of fusion architectures is the most popular in the literature due to the fact that centralized fusion can be characterized as a well defined problem. Data collected from all the sensors are processed in a single central unit that performs the fusion task. This approach is optimal when there are no communication problems (bandwidth, noise) and the central unit has enough computational resources to perform fusion among data. Most fusion algorithms have been developed for the centralized fusion architecture, and many applications have been realized using this approach.

However, in recent years distributed and hierarchical approaches are becoming more popular, thanks to the the spreading of communication technology. Hierarchical fusion architectures are based on different layers of nodes: at the lowest layer, fusion nodes collect data from sensors to perform a first fusion process on these data, then they send their results to a higher layer of fusion nodes. Each of the higher layer nodes collects the results of fusion from lower layers and perform a different fusion process among them. The overall architecture can be seen like a tree where each node is a fusion node and the leaves of the tree are the sensors. For example, in [15] a hierarchical architectures is presented for the design of a monitoring system for a power plant. The architecture is made of two layers of fusion, a first fusion is performed among a subset of sensors, then fusion results are send to the central monitoring system that fuses the results of the sensor subsets. This design allows to send to the central monitor more reliable and smaller data, resulting in an increase in the performance of the whole monitoring system.

Finally, distributed architectures differ from hierarchical ones in the topology of the fusion nodes. Also in this case each node performs locally a fusion process and send the results to other fusion nodes. However, in distributed architectures there are no hierarchical layers, but every fusion node can communicate with each other. The connections are thus arbitrary and the overall architecture can be represented as a graph of fusion nodes. A distributed approach to information fusion is very frequent in agent base systems, see for example [13, 16], that are presented in the next section.

As compared with the centralized ones, distributed and hierarchical architectures have the following advantages:

- Lighter processing load at each fusion node, due to the distribution over multiple nodes.
- Lower communication load, due to the reduction amount of data to be communicated.
- Faster user access to fusion results, due to reduced communication delay.

On the other hand, using distributed or hierarchical architecture requires the development of fusion algorithms that are specialized for those architectures.

In the rescue context, centralized approaches are not suitable, since the hypotheses of perfectly reliable and low-cost communication among the nodes (that in our case are agents acting in the rescue scenario) is not verified. In fact, during rescue operations often communication among the agents is very difficult, noisy, and with low bandwidth and thus a centralized approaches would easily lead to a complete stall of the operations of the agents.

On the other hand, using distributed or hierarchical approaches leads towards the use of multi-agent based approaches to perform the fusion and these approaches are seen best suited for the rescue domain.

Before defining our proposal for the design and development of cognitive agents for rescue operations, we have studied many multi-agent systems for information fusion that can be found in the literature. We report in the next section some of them, highlighting the features that may be of interest in our application domain.

3.1 Multi-Agent approaches to Information Fusion

In the last ten years, the multi-agent technology has became a very important research topic and many researchers in the information fusion field started exploring the possibilities to develop information fusion systems by making use of multi-agent systems. In the following we analyze different works which exploit the agent technology for the information fusion problem.

An important part of the multi-agent system techniques applied to the information fusion problem is the gathering and selection of information obtained as a consequence of a query, for instance to an Internet search engine. Even if the process of fusion is more related to the symbolic level, and thus it is not strictly a sensor fusion problem, this kind of works cover a large portion of the efforts of the information fusion research. The issues which arise while executing queries is to gather relevant heterogeneous results in a coherent manner avoiding their overlapping, and to deal with information sources which are not homogeneous in sharing a common ontology. The work of Zhang and Zhang [21] is one example of this kind of study: the authors present an agent based information fusion system which is used to collect results of the same query from different resources, and to address the decision fusion problem which arise when many agents make different decisions starting from the same information depending on their own knowledge base. The interaction between decisional agents and information retrieval agents described in this work is an interesting issue that we need to integrate in an architecture in the rescue domain, since also in this scenario there are agents that collect information about the status of the world, namely people in the disaster area, and agents that make decisions based on information available.

Aside this class of works, the multi-agent techniques has been applied to multi sensor system to fuse both raw sensor data and feature data. In [13] Knoll and Meinkoehn proposed an agent system architecture to realize a distributed sensor network. They started by analyzing the benefits which a well suited multi-agent system could bring to the development of a distributed sensor network by reducing the amount of data transferred among the sensors and a central fusion unit, and letting the sensor network be easy to expand. The solution they propose is a fully distributed sensor-agent network where every element has the capability to exchange data only with those network nodes which can really contribute to increase the global knowledge by providing useful data. This kind of distributed sensor networks may be useful in rescue domains when it is possible to exploit information of sensors in the environments (e.g. temperature sensors for detecting fires, video cameras for acquiring information about collapsed building or road traffic, etc.) and it is required to integrate this data with higher level information coming from people.

The work of Anderson [1] is focused on a smart use of a distributed sensor network by making use of a multi-agent system. It analyzes the advantages that agents can introduce in the Baltic Watch project framework. The goal of this project is to develop a monitoring structure which can increase the security condition within the Baltic Sea. Also in this work there is an important element to be considered for rescue operations that is the choice of the best communication topology among the agents. In fact, in a rescue scenario, due to the general difficulty in communicating with other agents, it is very important to define communication infrastructures and to ensure that each agent can communicate only with those agents that are able to effectively process and interpret the message.

Finally, the work of Regis et al. [16] is based on the same view of agent which extends a sensor. Their research investigates how a distributed real-time control system can be realized by making use of agents which control distributed sensors and execute the fusion process. They developed agents which have the capability to schedule the different tasks they needed to achieve their goal under soft real-time constraints. The importance of this work in rescue application is the introduction of the autonomy of each agent. In fact, every agent acting in a rescue domain must act in collaboration, but autonomously with respect to the others, in other words even if communications are not available for a period of time the agent must continue his work.

We have specifically considered all these works in order to define the system architecture for our fusion system in the rescue domain that is presented in the next section.

4 Agent design

In this section we propose an agent architecture that is suitable for the modeling of agents with the features required in a search and rescue scenario. Moreover, we define the components of a framework that allows for an easy and structured implementation of agents operating in real-time.

4.1 Agent model

The literature on agent approaches to information fusion shows that agents can be seen both as system components that allows for a modular software approach to system design and modeling concepts. We follow the latter approach thus identifying our system agents with the entities that act in the scenario.

The overall structure of the agent should satisfy the needs for information acquisition as well as the support for intelligent action. In Figure 1 we describe the functional model of the agent.

We sketch the main function accomplished by each process (represented as an oval in Figure 1).

World and World Management - The world is the structure that represents the whole agent knowledge about the world. In order to grant consistency it can only be modified by the *World Manager*, that implements different update policies. For instance, in the rescue domain the world is composed by the rescue world objects such as roads, agents, buildings, etc., plus some information about the agents state.

At each instant the world contains the result of the fusion of the information received since the starting time, and, hence, it contains the knowledge for decision making.

Plan Executor (PE) - A plan may be seen as a graph that specifies the actions to reach a goal. Each node is a state, edges specifies the state transitions caused by action

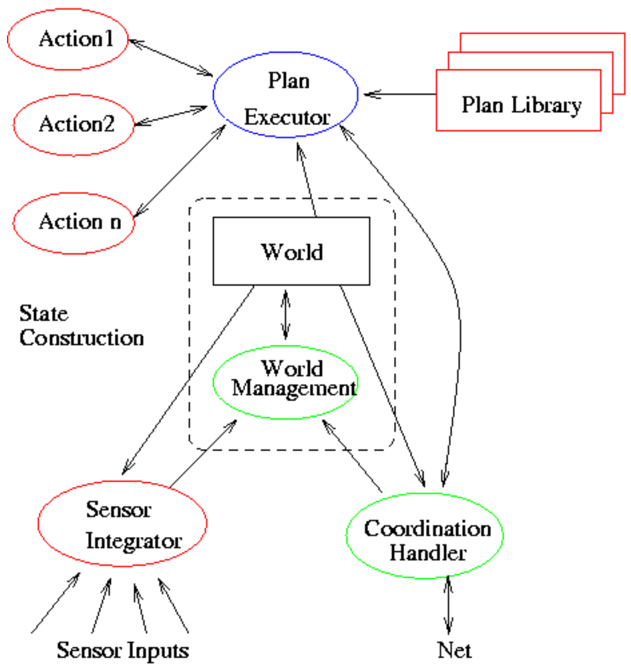


Figure 1 : Functional Agent Description

execution. The Plan Executor performs two tasks:

- It receives from the coordination handler the goal to reach and peeks from the plan library the plan to achieve it.
- It executes a plan by starting or stopping primitive actions at each state, finding which edge to follow in case of a condition branch.

A plan switch can occur after the recognition of a plan failure and/or a goal switch may be forced by the coordination module.

Sensing Integrator - In general, an agent can be equipped with a large variety of sensors and/or sensing capabilities, each one providing different type of input. The task of the sensor integrator is to calculate the new state of the world starting from the previous states and the current sensor info. At present, in the RoboCup-Rescue simulator, agents have limited sensing capabilities, modeling vision and hearing, and not requiring the application of sophisticated integration techniques.

Coordination Handler - In the multi-agent system the communication capability plays a fundamental role. Basically two kinds of information have to be handled:

- information about the sensed world
- information about the action coordination

The information exchanged by the agents about their perception of the world serve as input for the information fusion process. As already noted information can be represented by features or more generally and by symbolic representations. The information concerning the actions coordination is used to decide the *agent* specific goal, allowing the agent system to accomplish the overall goal of agent team.

As argued in the previous section information fusion can be accomplished at several degrees of abstraction and therefore the agent architecture must be heterogeneous in that it should handle different representations. In particular, we aim at combining fusion of raw sensor data (although in the simulator perception is currently not considered at this degree) as well as features and symbols. As shown in Figure 1, through the communication channel agents can exchange information about the world as perceived, and therefore regarded as additional sources of information that must be considered in order to update the representation of the world. In addition, the types of agents range from the operation center to the individual agent, possibly organized in several independent structures. This makes it necessary to provide centralized fusion capabilities as well as hierarchical and fully distributed ones.

In this respect, the proposals that have been presented in the previous section are generally focussed on one specific type of representation, as on a specific architecture, and therefore are not directly applicable in our scenario. To provide the required flexibility, the heterogeneous and asynchronous architecture proposed in [8] for the design of a single agent is adopted to implement a multi-agent system with the capabilities of a distributed organization for fusion and of the integration of information at different degrees of abstraction.

In fact, in this architecure heterogeneity allows for taking into account different level of representation of information acquired by the agents and asynchronicity allows for integrating planning capabilities and fusion processes in a dynamic environment.

4.2 ADK for Robocup Rescue

The Robocup Rescue environment provides a tool (the Rescue ADK [2]) that simplifies the agent construction. However, at present, the above sketched agent model cannot be directly defined on top of the ADK. Therefore, we are designing an extension of the ADK that is suitable for our purposes, by implementing the modules discussed above within our asyncronous architecture.

Moreover, we are concerned with the communication capabilities of the agents. In particular, we are interested in experimenting with communication infra-structures that cannot be directly modeled in the setting supported by the ADK. For example, one cannot specify that a set of agents is linked through a broadcast communication channel.

As specified by the ADK, agents have their own representation of the state (memory), but it does not allow to maintain multiple or uncertain representations of data, which are possibly needed to handle the fusion of data coming through the communication with other agents, and are required by our heterogeneous architecure.

Finally, we are implementing a mechanism for visualizing not only the global scenario, but also each agent view of the world in order to evaluate the process of situation assessment within the multi-agent system.

5 Conclusions

We have presented the current development of the ongoing project "Monitoring and Information Fusion for Search and Rescue Operations in Large-scale Disasters". The aim of the research is to develop a tool to support search and rescue operations in large scale disasters. In particular, we are deploying the RoboCup Rescue Simulator, which has been developed to provide an environment for experimentation of multi-agent technology in the framework of the RoboCup initiative. We have addressed a specific application domain: the disaster scenario recorded after the earthquake of Umbria and Marche. Moreover, we have focussed on the problem of situation assessment and discussed the features of some approaches to information fusion based on multi-agent approaches, that are related to the present project. Finally, we have sketched the agent model that we are developing in order to design the various types of agents.

The availability of the RoboCup-Rescue simulator has been extremely valuable for the development of the present project, providing an experimental setting that can be effectively used for developing a prototype implementation. The simulation can serve both to evaluate various strategies for information acquisition and situation assessment, as well as to make it more understandable to the VVF personnel the potential benefits of an integrated approach to the simulation and monitoring of a real search and rescue scenario. While it is premature to consider the effectiveness of the tool in the management of operation, both the analysis of past scenarios as well as the training of personnel seem to be already suitable for application.

Acknowledgements

This research was supported under contract number F61775-01-WE-030 of the European Office of Aerospace Research and Development, Air Force Office of Scientific Research, United States Air Force Research Laboratory.

We also gratefully acknowledge the technical support of the RoboCup Rescue Project.

References

[1] L. Andersson. Intelligent agents. *Master Thesis in informatics*, 1999.

[2] Micheal Bowling. Robocup rescue: Agent developement kit version 0.4, available at [17].

[3] P.R. Cohen, H.J. Levesque, and I. Smith. On team formation. *Contemporary Action Theory*, 1999.

[4] Robocup Rescue Technical Committe. *Robocup Rescue Simulator Manual v0.4*, available at [17].

[5] Pianificazione e monitoraggio in tempo reale dei soccorsi in gravi disastri (in Italian). http://www.dis.uniroma1.it/~rescue.

[6] B.J. Grosz. Collaborative systems. *AI Magazine*, 1996.

[7] D.L. Hall and J. Llinas (Eds.). *Handbook of Multisensor Data Fusion.* CRC Press, 2001.

[8] Luca Iocchi. *Design and Development of Cognitive Robots.* PhD thesis, Univ. "La Sapienza", Roma, Italy, 1999.

[9] H. Kitano and et al. Robocup, a challenge AI problem. *AI Magazine, Spring*, 1997.

[10] H. Kitano and et al. Search and rescue in large scale disasters as a domain for autonomous agents resarch. *IEEE International Conference on System, man and Cybernetics (SMC99)*, Tokyo, 1999.

[11] H. Kitano and et al. Wearable system for the disaster mitigation problem: Mission critical man-machine interface for the robocup rescue simulator. *Proceedings of International Conference on Artificial Reality and Telexistence(ICAT99)*, 1999.

[12] H. Kitano and S. Tadokoro. Robocup rescue, a grand challenge for multiagent and intelligent systems. *AI Magazine*, 2001.

[13] A. Knoll and J. Meinkoehn. Data fusion using large multi-agent networks: an analysis of network structure and performance. *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, (MFI '94)*, pages 113 - 120, 1994.

[14] Murphy, J.G. Blitch, and J.L. Casper. Robocup urban search and rescue events: Reality and competition. *AI Magazine*, In press.

[15] Du Qingdong, Xu Lingyu, and Zhao Hai. D-s evidence theory applied to fault diagnosis of generator based on embedded sensors. *Proceedings of the Third International Conference on Information Fusion (FUSION 2000), Volume 1*, 2000.

[16] B. Horling, V. Lesser, V. Regis, W. Thomas. The soft real-time agent control architecture Technical Report 02-14, University of Massachussets, 2002.

[17] Robocup Rescue Web Site.
http://robomec.cs.kobe-u.ac.jp/robocup-rescue.

[18] K. Sycara. Multiagent systems. *AI Magazine*, 19(2):79-92, 1998.

[19] S. Tadokoro and et al. The robocup rescue project: a multiagent approach to the disaster mitigation problem. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA00), San Francisco*, 2000.

[20] L. Valet, G. Mauris, and P. Bolon. A statistical overview of recent literature in information fusion. *IEEE Aerospace and Electronics Systems Magazine*, 16(3):7-14, 2001.

[21] Zili Zhang and Chengqi Zhang. Result fusion in multi-agent systems based on owa operator. *Proceedings of the 23rd Australasian Computer Science Conference (ACSC 2000)*, pages 234 - 240, 2000.

Structure and Working of the Rescue Simulator

A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, and M. Salerno

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113 - 00198 Roma, Italy
`last_name@dis.uniroma1.it`

Abstract. Due to the improvement of research in the autonomous agent field, it is becoming possible to simulate complex or critical activities, such as rescue operations in areas where large disasters occurred. The RoboCup Rescue Project aims to study the rescue issue under many different points of view (resource allocation, study of efficient procedures to coordinate the available involved rescue operators, etc.). To accomplish our goal, we have realized a framework to define a simulation environment that models the real environment, and within which the software agents can operate, based on the RoboCup Rescue Simulator [Com00]. In this document we first describe the main features of the RoboCup Rescue Simulator and then the installation done at our Lab.

1 Objectives

While developing the simulator the following objectives were considered:

- to describe an environment in terms of a generic map, composed of the most common urban elements:
  - building
  - roads

  Some geometrical and logistic properties are associated to each one of these elements, in order to model the reality with enough accuracy.
- to simulate different kinds of disasters, by supplying the possibility to develop simulation components in a modular manner, that is, in a way independent of the rest of the framework.
- to model agents which interact with the environment in an ad-hoc manner, and to supply a standard communication interface within the entire structure.

2 Implementation issues

To realize the framework described in the previous section it, is necessary to adopt a distributed model as a consequence of the high computational costs required by the process.

The system is divided in many components each of which can be executed on a different host. A reliable datagram communication protocol, based on UDP, has been adopted. At every simulation frame, only the state changes are sent, so that the network traffic is reduced.

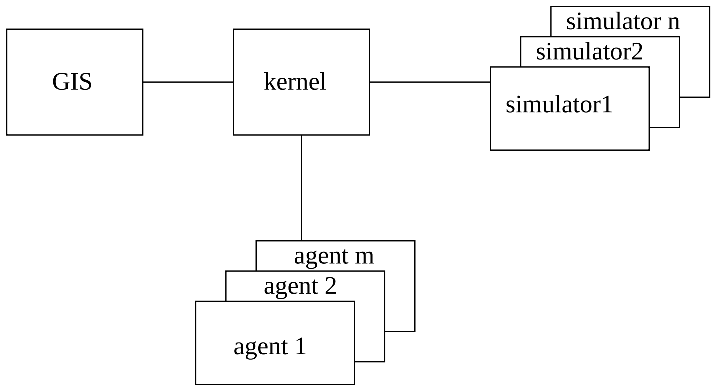
3 System structure

The simulation is realized in the discrete time system framework, where the state evolution in a certain instant is calculated on the basis of the state in the preceding instant, and of the inputs (represented by the agents/action).

The sequence of actions forming a simulation frame is the following:

- State representation
- Catastrophic event simulation
- Simulation results integration
- Inter-module communication
- State information transmission

module	*features*
GIS	state representation
kernel	simulation result integration, inter-module communication
misc sim traffic sim block sim fire sim collapse sim	simulates crowd, traffic fire, collapses
agents	civilian, police, ambulance, fire brigade



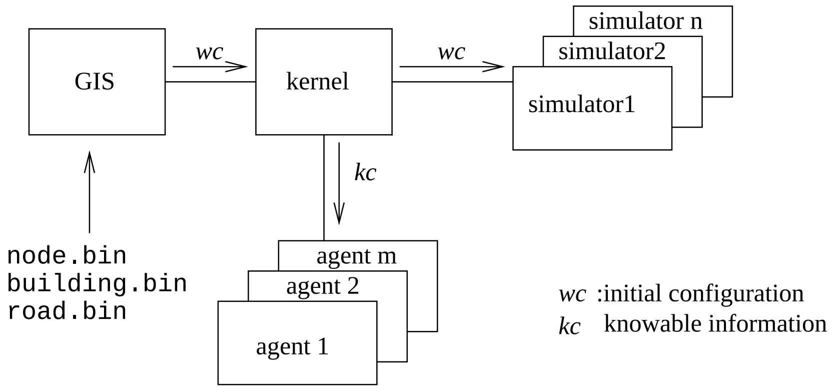
4 Execution

The system is based on a discrete time model. The information related to the state that is exchanged by the modules, is focused on objects, each of which is identified by an *id* attribute, and a certain amount of other properties. The object updating process implies only the retransmission of those properties that changed with respect to the preceding instant.

4.1 Initialization

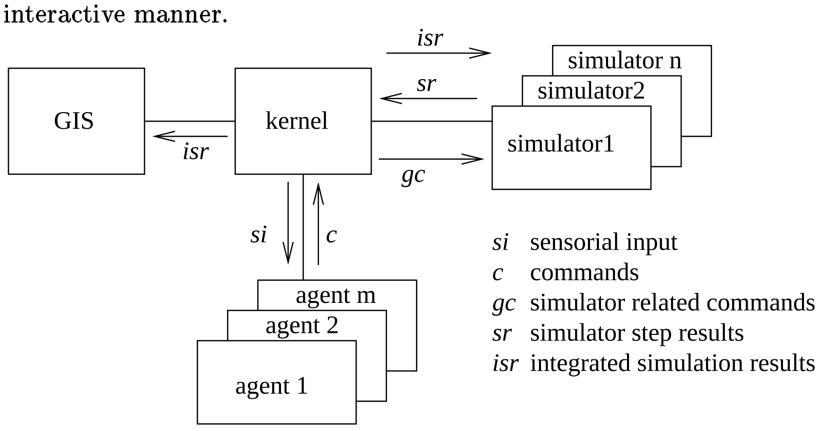
The data transmission technique adopted involves the necessity to maintain a copy of each module state, for this reason there is an initial phase where every module receives the state representation. The following steps are involved:

1. the environment state is loaded by the GIS from binary files
2. the kernel connects to GIS to obtain the entire state representation
3. the simulator connect to the kernel to receive the entire state representation
4. the agents connect to the kernel to receive a partial state representation



4.2 System behavior

Once the state has been reconstructed by the agents, the simulation proceeds in an interactive manner.



The sequence of actions executed in a generic step is:

1. the kernel sends the sensor information to the agents
2. every agent sends the action it intends to execute to a specific simulator via the kernel
3. the kernel sends the agents commands to the simulators
4. every simulator calculates the next state, then it sends the results to the kernel
5. the kernel integrate the received results, updates the GIS and increases the simulation time

Simulators While integrating information, it is assumed that whether a simulator changes a property, it is the only one allowed to. thus, the integration is simply obtained by merging the partial results received by the different simulators. Each simulator's result is integrated in the simulation instant sequent to the one in which it has been transmitted.

5 Installation

5.1 Set up

System Requirements:

– At least 192 MB of RAM.
– At least a linux 2.2.14 kernel
– Al least gcc-2.91.66
– jdk-1.3

What to do:

- unpack the `rescue-sim.tgz` archive in your home
  ```
  # tar -xzvf rescue-sim.tgz
  ```
  this step will add a directory named `rescue-sim`

- change directory to `rescue-sim`
  ```
  # cd rescue sim
  ```
- build the sources
  ```
  # make
  ```

The last step may take a very long time. If the compiler throws an internal error it is possible the system runned out of memory. The problem may be solved by killing some system service and restarting the build stage.

5.2 Running

To start a sample session of the simulator on the local machine just type
```
# ./all-sh
```
within the `RUN` directory.

The Rescue System is made up by the GIS [1], the kernel, a set of disaster simulators, the civilians and your agents. Each module is an independent process communicating with the others through the network. For this reason, if a large set of calculators is available, it is possible to spread the simulation over there, simply spreading the processes.

By installing a non interactive remote shell program, such as `rsh`, it is possibile to write a simple shell script that starts the processes in a distribuited way. Obviously the processes have to be informed on which machine the Rescue Kernel [2] is running; this can be done by setting some parameters on the command line. Since those parameters are undocumented we provide a short reminder extracted from the sources.

Parameters:

[1] Geographic Information System.
[2] The process that handles communication.

module	**parameters**
gis	(-file ConfigFile [3])
kernel	(-file ConfigFile)((-nogis)—(GISHost))
miscsimulator	(-file ConfigFile)(KernelHost)
collapsesimulator	(-file ConfigFile)(KernelHost)
firesimulator	(-file ConfigFile)(KernelHost)
collapsesimulator	(-file ConfigFile)(KernelHost)
blockadessimulator	(-file ConfigFile)(BlockParam)(KernelHost)
trafficsimulator	(-file ConfigFile)(BlockParam)(KernelHost)
samplecivilian	(-file ConfigFile)(KernelHost)
Civilian1	(-f ConfigFile)(-h KernelHost)(-n NumberOfAgents)
Civilian2	(-f ConfigFile)(-h KernelHost)(-n NumberOfAgents)
Civilian3	(-f ConfigFile)(-h KernelHost)(-n NumberOfAgents)

Sample batch file for distribuited execution:

```
#! /bin/sh

#here we assume to spread the simulation over three hosts

DISPLAY=$1
./0_rgis.sh $DISPLAY&
sleep 5
./1_rkernel.sh $DISPLAY $HOSTNAME&
sleep 5.
./kuwataviewer.sh -l 300 -h $HOSTNAME&

#set the remote host path
CD=/home/rescue/src/rescue-sim/RUN

#starts the simulator pool on host myri102
rsh myri102  $CD/3_rmiscsimulator.sh $DISPLAY $HOSTNAME&
rsh myri102  $CD/4_rtrafficsimulator $DISPLAY $HOSTNAME&
rsh myri102  $CD/5_rfiresimulator.sh $DISPLAY $HOSTNAME&
rsh myri102  $CD/6_rblockadessimulator.sh $DISPLAY $HOSTNAME&
rsh myri102  $CD/7_rcollapsesimulator.sh $DISPLAY $HOSTNAME&
sleep 5

#starts agents on host myri102
rsh myri103  $CD/rCivilian1.sh $DISPLAY -h $HOSTNAME&
rsh myri103  $CD/rCivilian2.sh $DISPLAY -h $HOSTNAME&

sleep 5
# replace YabAI with your agents!
xterm -e ./YabAI.sh myri101&
```

References

[Bow00] Micheal Bowling. Robocup rescue: Agent developement kit. 2000.

[Com00] Robocup Rescue Technical Committe. Robocup rescue simulator manual v0.4. 2000. http://robomec.cs.kobe-u.ac.jp/robocup-rescue.

Information Fusion

A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi, and M. Salerno

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113 - 00198 Roma, Italy
last_name@dis.uniroma1.it

Abstract. The problem of information fusion can be faced at different levels within the software architecture of a complex system. In the last twenty years information fusion has thus been interpreted by treating information in different ways: information as raw data, as objects features or as high level and complex structures.
In this paper we present a summary of the work on information fusion. We start by characterizing the notion. We then look at application domains, before focusing on the level at which information fusion takes place. We then address the architectures for information fusion, in particular agent-based approaches. We conclude by briefly addressing the task of information fusion in the rescue domain.

1 Information Fusion Characterization

During the last twenty years different meanings have been assigned to the wording *"information fusion"* (also referred to as *"data fusion"*), and all those related to them, depending on the diversity of the application domains to which the efforts of the corresponding researches were directed. In January 1998 a definition was adopted [Wald 99] which resumed the conclusions of an European working group, set up by the European Association of Remote Sensing Laboratories and the French Society for Electricity and Electronics (affiliate of the IEEE), concerning the specification of a lexicon of term of reference. We will adopt that terminology in this work, after having briefly explained and recalled the characterization of information fusion according to the following:

"Data fusion is a formal framework in which are expressed means and tools for the alliance of data originating from different sources. It aims at obtaining information of greater quality; the exact definition of 'greater quality' will depend upon the application."

Within this definition many terms like data fusion, information fusion, sensor fusion, classifier fusion are included, which means that information fusion can be done on many different types of data ranging, from raw measures to linguistic descriptions. Moreover, it is claimed that the architectures and the mathematical tools needed to execute the information fusion are part of the process and they are not the process itself. Finally, the link between quality and application field underlines that data fusion has to improve the value of the information obtained, because it satisfies more useful for the specific application which makes use of it, and not because it respects some strict predefined quality definition.

In this paper an overview of what information fusion is, when and how it has been applied and which are the directions it is heading to, will be presented, always referring to the above written definition.

2 Information Fusion Applications

Information fusion is broadly used in various application fields such as :

- Defense
- Geoscience
- Robotics
- Health
- Industry

The aim of this section is to present some recent works, concerning information fusion for each of the listed application fields, giving an intuition of how information fusion is becoming a very important and interesting research field.

2.1 Defense

Historically defense is the first field of application for information fusion. Lots of military application s involve the use of information fusion, such as aerial targets detection[5, 41, 49] identification[15] and tracking [3]
[23, 72], in [77] integration of a multi-source data fusion is performed for purpose of target tracking and identification with the existing Naval Command and Control for the HALIFAX frigates For purpouse of mine detection, information fusion from different kind of sensors (infrared camera, metal detector) is very effective and improves the probability of success[31]. In [28] a system using an innovative ground penetrating radar is used to recognize buried mines. Fusion is made among different data extracted from the GPR sensor; experiments are made on data collected on the battle field, a comparision is made between the results obtained using a primary algorithm and the result obtained performing the fusion: fusion actualy improves the performance of the overall system. Information fusion is also used in vehicle automatic detection [11, 40], battlefield surveillance, tactical situation assessment and object detection[1]. Finally, information fusion is also used to improve person authentication techniques, based on biometric measures (voice, face and profile images)[7, 12].

2.2 Geoscience

Geoscience concerns the earth study using satellite or aerial images [68, 88]. Fusion is used in Geoscience to detect areas of interest (rivers, mountains, airports, roads) merging images from different sources, or from different dates [51]. The main problem to achieve this goal is the classification and interpretations of images [38, 79]. In [46] merging of satellite images is used to improve the recognition process for planimetric features like roads energy transmission lines railroads or rivers. In [71] information fusion is performed with different aerial radar images to improve recognition of terrain areas. In [47] Map containing edge extracted from various source images (optical, infrared) are fused in order to obtain a combined edge map that has more reliable and more complete edge information. The combined map is then used to perform object recognition starting from the edge information. Results are presented starting from images coming from satellite. Another application of information fusion in Geoscience is the construction of a finer resolution image merging images of the same scene [10, 16, 18, 32].

2.3 Robotics

In Robotics information fusion is used for different purposes. Basically the problems to solve in this field of application are:

- Environment identification.
- Navigation and localization.

In [81], sensor fusion among different robot perception is used to improve the accuracy in estimating the position and tracking an object in the environment. The reported experiment show that the fusion process reduces the error on the object position. in others works fusion is used to improve the detection of outdoor environments as [61] where a robot is used to detect metal object hidden in the terrain. For navigation purpose information fusion is used in [85]. The navigation of an autonomous vehicle is approached using electrostatic potential fields and sensor fusion with fuzzy logic, information fusion is used to avoid obstacles. Experimental result on real show that the integration between fuzzy logic and electrostatic potential field results in shorter and safer paths respect to the use of the same techniques without fusion. Other works that use information fusion for navigation purposes are [30, 60, 63, 36, 42]. Lots of works approach the localization problem fusing information extracted from the envirnment and the information coming from odometry using a Kalman filter. In [84] the localization task of a mobile robot is solved fusing information from GPS and odometry (the robot is used as a guide in a campus). In [37] odometry is fused using the Kalman filter with the enviroment features extracted from an onboard camera. In the experiments, results obtained from the localization process with and without the fusion are compared, showing that fusion actualy improves the localization precision. Another work that uses information fusion for real-time localization is [9]. Fusion is also used to achieve temporal coherence between data in dynamic environments [39, 59].

2.4 Health

Health here is to be intended not only as human, but also in a more generic way as the well functioning of a system. In [52] techniques to improve aircraft engine

health are presented; a probabilistic approach to the fusion of data coming from multiple sources is used to improve the diagnostic and prognostic capabilities of a software system. In [74] information fusion is used to obtain a robust prediction detection system, the goal is to optimize the fusion system to have safer predictions and detections to achieve this different techniques of information fusion (such as Dempter-Shafter theory, Bayesian inference, fuzzy logic, neural network and simple weighting/voting)) are considered. Experiment results show the effectiveness of the fusion process.
In medical applications, information fusion is used for different tasks. In [20] information fusion is used for achieving outer wall detection of esophagus, from ultrasound images. Experimental results show that information fusion improve the outer wall esophagus detection. Other works where fusion is used for the modelization of the human body are [14, 62] In medicine information fusion is also used for tumor detection [78], using radiographic and ultrasonic images, classification of different tissue [62], bacterial recognition [89].

2.5 Industry

In the industry application information fusion is used to have a better control of the production quality. For example in [82] information coming from an electronic tongue and nose is fused to control the food quality. Information fusion is also useful for the automatization of production processes. In [50] sensor fusion is used to achieve a better object recognition. The use of fusion improves the correct identification of the objects and reduces the missing detection: in the article the percentage of object correct, false and missing recognition are reported. Information fusion is also used for automatic tool breakage detection [64], train localization or vehicle passage detection [83].

3 Fusion Techniques

Techniques used to perform information fusion are various and quite complex. It is possible to divide those techniques in four main categories [87]:

- Probability Theory
- Evidence Theory
- Fuzzy Set and Possibility Theories
- Neural Networks

The use of a specific technique in an application is due to many consideration such as:

- The level at which the fusion is performed. (signal, feature, symbol: see the next section)
- The constraint the application has to face (real time, off line)
- The architecture used to implement the application (centralized, distributed see Section 6)

- The kind (noisy, uncertain, etc) and amount of data the application needs to fuse

In this section we briefly discuss the difference between the techniques showing some example for each of them

3.1 Probability theory

Historically, probability theory is the first technique used to perform data fusion. Input data are modelled with probability or likelihood numbers, then data are merged using well known mathematical tools such as Bayesian rules or specific rules for the application at hand. In [81] an approach based on gaussian probability distribution is used to represent, fuse and communicate observations of an object by multiple robots. An ad hoc fusion technique is proposed to cope with gaussian distribution in a simpler and more efficient way. An example of information fusion performed using Bayesian rules can be found in [50], where fusion between multisensors is made for the recognition of object in an industrial environment using a Bayesian network. The different observations should not be considered independent in order to have better performance. Other works that use Bayesian methods are [7, 15, 38, 41]. The main drawback of the Bayesian approach is the identification of the a-priori probability distributions, moreover, to simplify computation, in most cases information sources are considered as independent: such an hypotesis is very restrictive in some cases.
Another well-known mathematical tool used to perform the fusion is the Kalman filter. The Kalman filter is based on the bayes' rule and provides a recursive estimation of the observed characteristics, starting from the previous observations and the known object characteristics. A large number of works uses this approach to perform object tracking [22] or to solve the localization problem in robotics applications[37, 81].

3.2 Evidence theory

Evidence theory is basically an extension of probability theory. Evidence theory is based on the initialization of basic belief assignment to the data sources; the belief assignments are merged togheter using rules, for example Dempster-Shafter rule. Allowing the handling of non-exclusive and non singleton events, Evidence theory is a powerfull tool in classification problems, where managing uncertainty is a main problem. For example in [75] the Dempster-Shafter theory is integrated with a geometrically inspired measurement of uncertainty, to fuse information from different kind of classifiers in a classification process, improving the reliability of the whole process. In [70] Dempster-Shafter theory is used to fuse multisensor data in monitoring system of power plant. As in probability theory the main problem for evidence theory is the initialization of the basic belief assignment. Efforts have been made to overcome this problem: in [53], a method for modelling the knowledge required for the initialization of the belief functions is presented. Experimental results are reported using real data, extracted from images of dermatological lesions. [54] presents a new approach to detemine automatically the mass function of the Dempster-Shafter theory. Another group of works propose the use of evidence theory togheter with fuzzy theory in the fusion process [21, 20].

3.3 Fuzzy Set and Possibility Theories

Fuzzy set theory is based on partial membership of element to sets. Membership functions describe the rate of membership of an element to a set, they are a very powerful interface between numerical and symbolic representations [91]. Fuzzy theory is also used to fuse numeric data, several defined operators allows for a very broad set of behaviors in the fusion of data. As example in [60] fuzzy logic is used both to fuse data from different regions of the same image, and to fuse signals from different sensors.
Another important aspect of fuzzy theory is the representation of uncertainty which is pursued by possibility theory. Possibility theory is proposed to deal with unprecise statement and to combine unprecise informations [9]. In [48] Fuzzy inference is used to manage uncertainty in the reasoning process.

3.4 Neural Networks

Neural Networks consist of layers of processing elements, that may be interconnected in various ways. Neural Networks are mostly used when the relationship between input and output data is unknown, because they can be trained on sets of data to obtain the desired behaviors [29, 49, 64, 86]. During the training process the neural network changes dynamically the function between input and output data, until the error on the training data sets is considered acceptable. Recently neural network have been used in the fusion process; for example in [82] a neural network approach is used to fuse information coming from an electronic tongue and nose, for food industry application. In [8] a Neuro-Fuzzy technique is adopted in classification of remote multisource sensing, and geographical data. The Neuro-Fuzzy approach results from the integration of fuzzy inference, neural network pattern recognition and derivative-free optimization techniques based on genetic algorithms. Neural networks and statistical approaches are normally considered as alternative techniques for data fusion; in [4] a study is reported to see whether the two approaches can be combined to have better performance.

4 Fusion Levels

The design and architecture of a fusion system is a very important and discussed issue in the literature concerning information fusion. The main concept of this issue is the level of fusion. Basically data can be fused at three levels (see [34]):

- Signal level
- Feature level

– Symbol level

Dasarathy in [19] spreads the levels of fusion into five. The five levels of fusion results from the input-output combination of the above three levels (Data input-Data output, Data input-Feature output, Feature input-Feature output Feature input-Decision output, Decision input-Decision output). The spreading of the levels according to Dasarathy leads toward a more flexible design of the fusion system architecture.
The choice of fusing data at a certain level is strictly related to the purpose and characteristics of data fusion and strongly influences the techniques adopted for the fusion. When the fusion is made at symbol level, the objects to be combined are a logical representation of data; therefore to fuse data at the symbol level in most cases logical operators are used. When the fusion is made at signal level, data are directly extracted from the sensors and no intermediate representation is given; the techniques used to fuse this kind of data are taken from signal theory. Same techniques, such as neural networks and fuzzy theory, can be used with very different data representations. Several efforts have been done to approach the problem of information fusion focusing on the level of fusion [19, 34, 68, 88], [25] presents a methodology for examining the interface between aircraft system/subsystem elements of a Prognostic and Health Management system. The methodology described perform the integration of different level of information fusion (Data, Feature, Information) in a coherent architecture. In the following, we represent an overview of the litterature on fusion, classifying the different approaches according to the level where fusion is performed.

4.1 Fusion at Signal Level

At the signal level, the fusion is made on data directly taken from sensors (see for example) [5, 18, 51]. At this level, no intermediate representation of data is used, the techniques commonly used are taken from signal theory and depend strictly from the kind of sensor. In some works, signal level fusion is used as part of a larger framework, data are aggregated at signal level, then the aggregated data are further fused at higher level (feature or symbolic) [50, 20].

4.2 Fusion at Feature Level

In the feature level data are extracted from sensors, and an internal representation is built for the fusion [14, 39]. We classify under the feature level those approaches that rely on a feature based internal representation that is specifically designed for the fusion purpose. The kind of feature chosen is strictly connected to the technique chosen for the fusion. Using the Dempster-Shafter approach to perform the fusion, an internal representation is needed in order to obtain the basic belief assignment (see previous section)
[53, 92, 21, 70]. In most cases the feature extraction is needed in order to avoid noisy data coming from the sensor's measurement. In the feature level fusion, techniques such as neural network [6], fuzzy rule [82] or a combination of them

[21] are very frequently used because these techniques are very flexible and can perform fusion obtaining good performance on several kinds of data representations. Other works use stochastic based fusion method working on ad hoc feature extracted from sensor data; in such cases, the feature to represent data is chosen in order to achieve some advantages in the fusion process. In [81], for example, the gaussian ditribution of object in the space is represented trough the parameter characterizing the function, in order to require less computation efforts in the fusion process.

4.3 Fusion at Symbol Level

At the symbol level, data are represented with logical constructs [41, 78]. In [45] the fusion is performed using operators between theories and models. Following this approach, coherence between data can be obtained, thus avoiding data inconsistency. The symbol level is the higher level of abstraction, hence the logical representation of data has to be extracted from sensor data through rather complex data processing: in [44], an architecture to perform a multisensor data fusion is presented. Other works that perform fusion at the symbolic level are [41, 78].

5 Fusion Architectures

Fusion architecture is becoming a very important issue in the design of a fusion system. The fusion architectures can be divided in:

- centralized
- hierarchical
- distributed

The centralized approach is the most popular in the literature due to the fact that centralized fusion can be characterized as a well defined problem. In recent year the distributed or hierarchical approaches are becoming more popular, thanks to the the spreading of communication technology.

5.1 Centralized

The centralized approach is the simplest one. Data collected from all the sensors are processed in a central unit that is the only one to perform the fusion;the results of the fusion process are send to the various users. This approach is optimal when there are no communication problems (bandwidth, noise) and the central unit has enough computational resources to perform the fusion among all data. Most fusion algorithms have been developed for the centralized fusion architecture, and many applications have been developed using this approach.

5.2 Hierarchical

Hierachical fusion architectures are based on different layers of fusion nodes. At the lowest layer, fusion nodes collect data from sensors to perform a first layer fusion among those data, then send their fusion results to a higher layer of fusion nodes. Each of the higher layer fusion nodes collects the results of fusion from lower layers and perform a second layer fusion among them. The overall architecture can be seen like a tree where each node is a fusion node and the leaves of the tree are the sensors. In [70] a hierarchical architectures is presented for the design of a monitoring system for a power plant. The architecture is made of two layers of fusion, a first fusion is performed among a subset of sensors, then fusion results are send to the central monitoring system that fuses the results of all the sensors subset. This design allows to send to the central monitor more reliability and smaller data, resulting in increased performance of the whole monitoring system. In [2] and [17] a hierarchical approach is used in a multiagent base system: such works are presented in the next section.

5.3 Distributed

Distributed fusion architecture is similar to the hierachical one. Is based on fusion nodes that perform locally the fusion process among the sensors, and send to other fusion nodes the fusion results. The difference is that there are no layers in these architectures, and every fusion node can communicate with each other. The connections are arbitrary, the overall architecture can be represented by a graph where every node is a fusion node and every edge represents a connection between the nodes. A distributed approach to information fusion is very frequent in agent base systems, see for example [43, 73], those approaches are presented in the next section.

As compared to the centralized ones, distributed and hierarchical architectures have the following advantages:

- Lighter processing load at each fusion node, due to the distribuition over multiple nodes.
- Lower communication load, due to the reduction amount of data to be communicated
- Faster user access to fusion results, due to reduced communication delay

On the other side, using distributed or hierarchical approaches one has to develop fusion algorithms that are specialized for those architectures. In [55] a tracking problem in a distributed and hierarchical architecture is faced. Some algorithms (JPDA,MHT) used to perform the fusion in tracking problem with centralized architectures are extended for distributed and hierarchical architectures. Using the distributed or hierarchical approaches leads towards the use of agentsbased approaches to perform the fusion. In the next sections we describe the advantages of using multiagents to perform fusion and present some recent works based on this approach.

6 Agents and Fusion

In the last ten years, the multi-agent technology has became a very important research topic. Agents are often considered by many authors, with respect to the software design process, as the descendants of the objects: they both encapsulate a state and rely on the concept of message exchanging. Developing software by making use of agent techniques is also referred to as Agent Oriented Programming [90]. It is generally agreed that there are some important features an agent must exhibit [67]. Before introducing the works an agent-based approaches to information fusion, we briefly summarize these features.

6.1 Agents and architectures

Agents As already mentioned, there is no generally accepted definition of agent. Many researchers defined agents very precisely, but agent definitions turn out to be very different [56]. In fact, agent definitions often embody the characteristic of a specific implementation. [67]. Anyway, most authors agree on a set of attributes an agent must possess. Among them [65, 26, 27, 57, 35, 90], the agent is to be "autonomous". Autonomy means that the agent has control over its actions and behaviors, it can act without external interference, both from users and other agents, and, finally, it can take initiatives to accomplish a given task.

The ability to cooperate with other agents and/or with users, both to achieve a given goal or to help someone else to accomplish its own task, is another generally recognized capability an agent has [65, 90, 26].

Two other features which are generally attributed to an agent are: the capability of learning from other agents, from the environment or even from users [26, 65, 90, 80], and the ability to "understand" the environment in which an agent works, thus acquiring and maintaining its knowledge about it [65, 26, 27, 57, 35, 90, 80]. Certainly there are other characteristics which can be used to define an agent, but it is difficult to identify a common basis: for a detailed discussion of each of them it is convenient to consult for example [65, 26, 27, 57, 35, 90, 80, 24].

Agent architectures The way an agent exhibits its autonomy depends on the fact that it can take decisions in response to some event. These responses are selected based on the different types of decisional processes which are used to implement the agent itself. The architecture of an agent characterizes these different processes. Substantially, there are three different architectures, which are e specially used in characterizing a robotic agent: the reactive one, the deliberative one, also called cognitive [24], and the hybrid architecture. The core of a deliberative architecture is the sense-plan-act model in which the environmental modifications are perceived and used to elaborate a plan to keep accomplishing a given goal. Once a plan, which is a sequence of actions the agent has to carry out, has been created, or more commonly selected from a set of previously built plans, the actions the plan itself involves are executed. The reactive architecture can be thought as the counterpart of the deliberative one, because it relies on the principle of action and reaction. In this case a set of reactive behaviors are included within the agent and as soon as an activating condition of one of these behaviors is triggered the agent executes it. Usually an agent has more than one behavior and a mechanism to execute the more relevant to the present situation is used [24, 58, 66, 13].

The hybrid architecture is a combination between the two opposite solutions which tries to take advantage from both of them, thus it results to be the most commonly implemented and developed.

6.2 Agent approaches to Information Fusion

In the last years, researchers in the information fusion field started exploring the possibilities to develop information fusion systems by making use of multi agent systems; in fact, there is a a strict relation between the sensor fusion problem domain, in particular in the case of a distributed sensor network, and a multi agent system domain. In the following we analyze different works which exploit the agent technology for the information fusion problem, to compensate one or more of the implicit issues which multi-sensing brings. The problems related with the possibility to easily expand or upgrade the sensor network; the communication issues which arise because the transmission of sensor data occupies a large portion of the available bandwidth, thus limiting the sensors' number, and because of lack of reliability of the communication itself; the overhead of the computational resources due to the fusion process of a large amount of gathered data.

An important part of the multi agent system techniques applied to the information fusion problem is the gathering and selection of information obtained as a consequence of a query, for instance to an internet search engine. Even if the process of fusion is more related to the symbolic level, and thus it is not strictly a sensor fusion problem, this kind of works cover a large portion of the efforts of the information fusion research. The issues which arise while executing queries is to gather pertinent heterogeneous results in a coherent manner avoiding their overlapping, and to deal with information sources which are not homogeneous in sharing a common ontology. The work of Zhang and Zhang [93] is one example of this kind of study. The authors present an agent based information fusion system which is used to collect results of the same query from different resources, and to address the decision fusion problem which arise when many agents make different decisions starting from the same information depending on their own knowledge base. Agent finalized to different tasks are realized (interface agents, decision making agents, information retrieval agents) and, in particular, information fusion agents which make use of the Ordered Weighted Averaging operator to implement the information fusion engine. Both the decisional agents and information retrieval agents send the results of the execution of their assigned tasks to the information fusion agents. All the agents make use of KQML as their communication language.

Aside this class of works, the multi agent techniques has been applied to multi sensor system to fuse both raw sensor data and feature data. In [43] Knoll and Meinkoehn proposed an agent system architecture to realize a distributed sensor network.

They started by analyzing the benefits which a well suited multi agent system could bring to the development of a distributed sensor network by reducing the amount of data transferred among the sensors and a central fusion unit, and letting the sensor network be easy to expand. The solution they propose is a fully distributed sensor-agent network where every element has the capability to exchange data only with those network nodes which can really contribute to the increase of knowledge by providing useful data. The core of their proposal is a Contract Network Protocol. The protocol allows for recruiting those sensor which satisfy the required data accuracy (sensors can be different among each others, thus having different reliability and resolution properties) and satisfy both time and space constraints. Subsequently Knoll et al. [76] realized a sensor-agent development framework called MagiC, which let the previously proposed architecture be easily realized. They prove the effectiveness of their solution by experimenting with an image acquisition system composed by four cameras used to drive a robotic harm in a lab environment. Finally, in [43] the computer vision problem of object recognition is successfully faced by using the architecture they propose, realized with the sensor-agent framework they developed. In this work the information fusion was exploited at the feature level.

The work of Anderson [56] is focused on a smart use of a distributed sensor network by making use of a multi agent system. It analyzes the advantages that agents can introduce in the Baltic Watch project framework. The goal of this project is to develop a monitoring structure which can increase the security condition within the Baltic Sea, by making the different states that are around it collaborate. One of the main points of the project is to gather information from all the available sensors (which can be real sensors, ship transponders or even human operators) which are active in that area to reconstruct, by fusing these data, an overall picture of the Baltic Sea. Because different sensors are working together, the author proposed to realize a sensor-agent for each of them with the capability to decide which of the other sensors should be activated to keep on monitoring for example ship. In this way, a smaller amount of better quality data would be generated and a better fusion process could be produced. Moreover, the researcher identified other kind of agents which could have a very deep impact over the system efficiency: agents which represent the ships involved in the scenario. In this manner ship-agents could help the sensor-agent network to keep a more strict control over the Baltic Sea area by letting it easily follow their movements.

The work of Regis et al. [73] is based on the same view of agent which extends a sensor. Their research investigates how a distributed real-time control system can be realized by making use of agents which control distributed sensors and execute the fusion process. They developed agents which have the capability to schedule the different tasks they needed to achieve their goal under soft real-time constraints. Each agent is in charge to select the most appropriate agent-sensor to collaborate with, depending on its reliability and on the quality of the information it can provide. The authors faced the problem by dividing the agent development in two different phases: the first goal was to implement within the agent, the technologies needed to reason about real-time such as, for instance, real-time scheduling technique, while the second goal was to optimize the running time of the adopted technologies. They also experimented the system they developed by using as a test bed a target tracking problem. They used many radar sensor-agents which had to triangulate the position of many moving targets within the environment. The data exchanging and the identification of the appropriate sensor to collaborate with, together with other tasks like, for instance, discovering new targets, had to be completed within one second to guarantee the validity of the acquired data used in the fusion process. The use of agents improved the quality of the global task and decrease the communication overload.

While some researchers have a vision of agents and sensors gathered in a single unit (we can talk about sensor-agent), others prefer to use the agents as supervisors of the distributed sensor network, often being in charge to activate or deactivate a specific sensor due to its characteristics. Ayari and Haton [2] realized a framework to cope with the sensor data fusion problem both at signal and feature level. They experimented it with one robot which builds and maintains a map of a structured environment (eventually subject to dynamic modifications) by making use of its sensors and a multi agent system to drive the entire process. The authors made use of two different classes of agents: a perceptual one and a interpretative one. The first, composed by five specialized subclasses, is in charge of the raw data analysis and the feature extraction process, while the second one, which is in turn composed by three different subclasses, has the control over the maintenance of the environmental model depending on the results of the perceptual agents computations. This kind of architecture lets the agents improve the quality of the data acquisition and map building tasks by selecting the appropriate sensor to activate and the more appropriate task to accomplish. The entire process makes use of the Contract Network Protocol to enable the agents interaction, but, as the authors themselves noticed, it suffers from the broadcast communication model that was adopted to realize it.

In the research work presented by Cozien et al. [17] agents are developed to work in conjunction with a neural network in a computer vision contest. They realized a system called Jarod, to recognize military airplane shapes from satellite pictures. Their system was composed by a pool of agents assigned with the task to extract the airplane shapes from an image. Once they complete their operations the results are passed to a neural network which is in charge to recognize the airplane type. The agents implemented are reactive ones, and their sensing behavior is driven by different types of gradient algorithms (Sobel, Prewitt, Laplace) or by ad hoc agent algorithm (the authors refer to each different algorithm with the term sensor). The agents have the ability to decide which is the most appropriate algorithm to use depending on the typology of the image they have to analyze. Once a shape has been identified it is passed to the neural network which relates it to an airplane model or, if the information obtained is not satisfying, it requires the creation of specific agents to refine, with a local analysis, the portion of image from which the shape has been extracted. The system they implemented resulted in a hierarchical one, where the neural network gathers the information, in this case represented by a set of features, obtained from the agents, while it is fully distributed in the feature extraction process (each multi agent system works on a different image running in a different workstation).

The last work we are going to analyze that concerns the possible use of multi agent system for information fusion technology, is made by Qi et al. [69]. Their research took a different direction as compared to the others because it makes use of mobile agents. Mobile agents are agents which can move around a network and execute locally where they believe their services are needed. To have a more detailed description of what is the so called Mobile Agent Paradigm, and which is the actual situation of this specific research field, see [33]. In the system proposed by the authors, agents incorporate a portion of the fusion engine and move around a sensor network analyzing the data the sensors produce; after a while, the agents come back to a main processing entity. The processing entity fuse all the information gathered by the agents and eventually communicates and fuse its results with those of other processing entities, present within the sensor network, to improve the global result quality. The result of this architecture is a lower workload of the communication structures obtained by reducing the amount of data transferred along the network. Moreover, the system increases its reliability with respect to a possible network failure which could avoid the data transmission, the agent indeed works locally and, as soon as the transmission are rearranged, it can move to a different node while maintaining all the data acquired in the meanwhile. The advantage of this system is in the usage of the computational resources which are involved in the fusion process that are consumed only when they are required.

7 Information Fusion in the Rescue Domain

Our interest in information fusion arose by analyzing a specific domain which is the management and planning of rescue operations in large scale disaster, such as, for instance, an earthquake. In such a situation the efficiency of aids is strictly dependent on the accuracy of the knowledge on the environment.

The level of information that it is possible to obtain in this context, is strictly dependent on the source generating the information. In a rescue scenario both signal level and feature level as well as symbol level information are present.

Signal level information is made of raw sensor data. Such a data could be camera images, radar scans, infraread sensor readings and so on. Feature level information is the one extracted after some processing of sensor data. Usually this kind of information is more compact than signal level one, and commonly it is related to a more complex data structure. Feature level information can be generated either by human beings or by an automated analysis of source data. An example of information at feature level can be the evaluation of a building burning degree, or the blocking level of a road.
Finally, symbol level is related with logic structured data that are suitable to perform automatic reasoning and among the three discussed levels is ofthen the more closely related to human common sense.
Symbol and feature level information are mainly needed within the decisional process, where the coordination of the rescue team is managed. In this contest symbolic knowledge is used to better allocate the available resources or to direct in a more effective manner the execution of a particular task.

Un the rescue domain, the architecture of the team is a very important issue. The fire departement for example has a hierarchical structure with three different layers: an operational unit where the main decisions are taken, a local unit which maintain radio communication with the operational unit and locally coordinate the work of the fire fighters and, finally, the fire fighters themselves who operate in the disaster area. Each of this layers executes a specific task and has to acquire and fuse information to take decisions.

The operational unit supervises the entire rescue operation and makes use mainly of information at symbolic and feature level. The feature level is built on the information provided by the local coordination units.

Each kind of unit takes decisions and act on the basis of both the information coming from lower and upper layers. Moreover each unit propagates its observations acting as an information source for the upper layer.
For instance, in this model the lowest layer is represented by the single fire fighter that acts on the basis of sensorially percepted situations and communicates with the local coordination unit. Overload of communication is avoided by making use of local fusion procedures to provide the upper level in the hierarchy only with the information which is relevant to the upper node task.

Information fusion is a central task in the rescue issues and, by analyzing the structure of a real fire department, it is possible to pinpoint how the information fusion is executed at different levels, with respect to the information type, and at different layers, concerning the hierarchical structure of the decisional/operational units. The use of a multi-agent system is, in our opinion, an obvious choice to realize the above mentioned autonomy in terms of both information fusion and task execution.

The RoboCup rescue simulation provides with a dynamically evolving simulated disaster environment, within which it is possible to implement software agents to simulate rescue teams. Realizing a proper communication simulator, enables the performance analysis of several communication infrastructures not

just in terms of communication bandwidth, including also communication problems that could arise in the particular domain, but also in terms of the effectiveness. In particular such a communication simulator allows both to represent the communication structure used by the fire department and to test different communication schemas.
Morover, in order to develop agents able to perform information fusion at every level, agents need a powerful comunication layer. For each agent it is also possible to encapsulate a fusion engine specific for the fusion level it has to deal with Moreover, even if not considered at the present time, a direct use of sensor data can be realized by developing sensor-agent which can be added to the system, thus letting us to investigate how to improve the usage of the signal fusion level.

Due to the autonomy which each agent has to exhibit while executing its assigned task, we believe that an hybrid architecture agent well fits the issues involved by rescue scenario. An high level reasoning layer manages the decision process by making use of both symbolic and feature information, while a reactive layer manages the reactions involved by unexpected environment modifications. In our work we will focus on the deliberative aspect of the decisional process, paying special attention to the feature fusion and symbol fusion level.

References

1. R.T. Antony. Database support to data fusion automation. *Proceedings of the IEEE*, pages 39 – 53, 1997.
2. I. Ayari and J.-P. Haton. A framework for multisensor data fusion. *Emerging Technologies and Factory Automation, 1995. ETFA '95, Proceedings., 1995 INRIA/IEEE Symposium on*, pages 51 – 59, 1995.
3. M. Aziz, Tummala M., and R Christi. Fuzzy logic data correlation approach in multisensor-multiagent tracking systems. *Signal Processing76(2)*, pages 195–209, 1999.
4. C. Barat, E. Loaiza, H.; Colle, and S. Lelandais. Neural and statistical classifiers. can such approaches be complementary? *Instrumentation and Measurement Technology Conference, 2000. IMTC 2000. Proceedings of the 17th IEEE Volume 3*, pages 1480 – 1486, 2000.
5. A. Bastiere. Methods for multisensor classification of airborne targets. *Aerospace Sience and Tecnology, (1)*, pages 401–411, 1998.
6. M.D. Bedworth. Source diversity and feature-level fusion. *Information, Decision and Control, 1999. IDC 99. Proceedings.*, pages 597 – 602, 1999.
7. S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz. Fusion of face and speech data for person identity verification. *Neural Networks, IEEE Transactions on*, pages 1065 – 1074, 1999.
8. J.A. Benediktsson, H. Benediktsson, and K. Arnason. Absolute neuro-fuzzy classification of remote sensing data. *Geoscience and Remote Sensing Symposiun, 2000. Proceedings. IGARSS 2000. IEEE 2000 International Volume: 3*, pages 969 – 971, 2000.
9. P Bison, G. Chemello, C. Sossai, and G. Tranito. A synthetical approach to data fusion. *Lecture notes in Computer Science(1244)*, pages 58–70.
10. P. Blanc, T. Blu, T Ranchin, L. Wald, and R. Aloisi. Using iterated rational filter bakns within the arsis concept for producting 10m landsat multispectral images. *International Journal of Remote Sensing 19(12)*, pages 2331–2343, 1998.
11. D. Borgheys, P. Verlinde, C. Perneel, and M. Acheroy. Multilevel data fusion for the detection of targets using multispectral image sequences. *Oprical Engeneering, 37(2)*, pages 477–484, 1998.
12. D. Borgheys, P. Verlinde, C. Perneel, and M. Acheroy. A method for still image interpretation relying on a multi-algorithms fusion scheme, application to human face characterization. *Fuzzy Sel and Systems, 103(2)*, pages 317–337, 1999.
13. R. Brooks. A robust layered control system for a mobile robot. **IEEE Journal of Robotics and Automation*, Vol. RA2, April*, page 1423, 1986.
14. R.P. Broussard, S.K. Rogers, M.E. Oxley, and G.L. Tarr. Physiologically motivated image fusion for object detection using a pulse coupled neural network. *Neural Networks, IEEE Transactions on Volume: 10 Issue: 3*, pages 554 – 563, 1999.
15. D.M. Buede and P. Girardi. A target identification comparison of bayesian and dempster-shafer multisensor fusion. *Systems, Man and Cybernetics, Part A, IEEE Transactions on Volume 27 Issue5*, pages 569 – 577, 1997.
16. M. Costantini and F. Farina, A.and Zirilli. The fusion of different resolution sar images. *Proceedings of the IEEE Volume: 85 Issue: 1*, pages 139 – 146, 1997.
17. R. Cozien, C. Rosenberger, P. Eyherabide, J. Rossettini, and A. Ceyrolle. Target detection and identification using neural networks and multi-agents systems. *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, 2000.

18. M.M. Daniel and A.S. Willsky. A multiresolution methodology for signal-level fusion and data assimilation with applications to remote sensing. *Proceedings of the IEEE Volume: 85 Issue: 1*, pages 164 – 180, 1997.
19. B.V. Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE Volume: 85 Issue: 1*, pages 24 – 38, 1997.
20. R. Debon, B. Solaiman, M. Robaszkiewicz, and C. Roux. Data fusion and stochastic optimization: application to esophagus outer wall detection on ultrasound images. *Engineering in Medicine and Biology Society, 2000. Proceedings of the 22nd Annual International Conference of the IEEE Volume: 2*, pages 1439 – 1442, 2000.
21. T. Denoeux. A neural network classifier based on dempster-shafer theory. *Systems, Man and Cybernetics, Part A, IEEE Transactions on Volume: 30 Issue: 2*, pages 131 – 150, 2000.
22. Dietl, J. S. Gutmann, and B. Nebel. Cooperative sensing in dynamic environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, 2001.
23. J.A. Fayman, P. Pirjanian, H.I. Christensen, and E. Rivlin. Exploiting process integration and composition in the context of active vision. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on Volume 29 issue 1*, pages 73 – 86, 1999.
24. J. Ferber. Multi-agents systems. *Addison-Wesley:Reading,MA*, 1999.
25. B.L. Ferrell, J.L. Cruickshank, B.J. Gilmartin, C. Massam, S.J.and Fisher, and F.D. Gass. Case study methodology for information fusion interface definition. *Aerospace Conference, 2001, IEEE Proceedings. Volume: 6*, pages 3003 – 3015, 2001.
26. L. N. Foner. What's an agent anyway ? *Proceedings of the First International Confrence on Autonomous Agents*, pages 1–40, 1997.
27. S. Franklin and A. Graesser. Is it an agent or just a program? a taxonomy for autonomous agents. *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, Springer Verlag*, 1996.
28. P.D. Gader, J.M. Keller, and B.N. Nelson. Recognition technology for the detection of buried land mines. *Fuzzy Systems, IEEE Transactions on*, pages 31 – 43, 2001.
29. P.D. Gader, M. Mohamed, and Jung-Hsien Chiang. Handwritten word recognition with character and inter-character neural networks. *Systems, Man and Cybernetics, Part B, IEEE Transactions on Volume: 27 Issue: 1*, pages 158 – 164, 1997.
30. P. Gamba, R. Lodola, and A. Mecocci. Scene interpretation by fusion of segment and region information. *Image and Vision Computing, 15(7)*, pages 499–509, 1997.
31. E. Gelenbe, T. Koncat, and L. Collins. Sensor fusion for mine detection with the rnn. *Lecture Notes in Computer Sience, (1327)*, pages 947–942, 1997.
32. D. Granrath and J. Lersch. Fusion of images on affine sampling grids. *Journal of the Optical Society of America, 15(4)*, pages 791–801, 1998.
33. R. Gray, D. Kotz, G. Cybenko, and D. Rus. Mobile agents : Motivation and state of the art systems. 2000.
34. D.L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, pages 6 – 23, 1997.
35. B. Hayes-Roth. An architecture for adaptive intelligent systems. *Artificial Intelligence: special issue on agents and interactivity, 72*, pages 329–265, 1995.
36. Hu-Guohui and Yang-Chunjun. Study of multisensor information fusion filter in integrated navigation. *Transactions of Nanjing University of Aereonautics and Astronautics, 14(2)*, pages 122–125, 1997.

37. Luca Iocchi, Domenico Mastrantuono, and Daniele Nardi. A probabilistic approach to Hough Localization. In *ICRA01*, pages 4250–4255, 2001.
38. Byeungwoo Jeon and D.A. Landgrebe. Decision fusion approach for multitemporal classification. *Geoscience and Remote Sensing, IEEE Transactions on Volume: 37 Issue: 3 Part: 1*, pages 1227 – 1233, 1999.
39. R. Joshi and A.C. Sanderson. Minimal representation multisensor fusion using differential evolution. *Systems, Man and Cybernetics, Part A, IEEE Transactions on Volume 29 Issue 1*, pages 63 –76, 1999.
40. E. Jouseau and B. Dorizzi. Neural network and fuzzy data fusion application to an on-line and real time veichle detection system. *Pattern Recognition Letters 20*, pages 97–107, 1999.
41. M. Kam, C. Rorres, Wei Chang, and Xiaoxun Zhu. Performance and geometric interpretation for decision fusion with memory. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, pages 52 – 62, 1999.
42. M.and Xiaoxun Zhu Kam and P. Kalata. Sensor fusion for mobile robot navigation. *Proceedings of the IEEE Volume: 85 Issue: 1*, pages 108 – 119, 1997.
43. A. Knoll and J. Meinkoehn. Data fusion using large multi-agent networks: an analysis of network structure and performance. *Multisensor Fusion and Integration for Intelligent Systems, 1994. IEEE International Conference on MFI '94.*, pages 113 – 120, 1994.
44. M. Kokar and Z. Korona. Model based fusion for multisensor target recognition. 1996.
45. M. Kokar, A. Tomasik, and J. Weyman. A formal approach to information fusion.
46. Pigeon L., Solaiman B., Toutin T., and Thomson K. P. B. Dempster-shafer theory for multi-satellites remotely-sensed observations.
47. Yiyao L., Venkatesh Y. V., and Ko C. C. A knowledge-based neural network for fusiing edge maps of multi-sensor images.
48. D.A. Lambert and M.G. Relbe. Reasoning with tolerance. *Knowledge-Based Intelligent Electronic Systems, 1998. Volume: 3*, pages 418 – 427, 1998.
49. G.A. Lampropolus, V. Anastassopolus, and J.F. Boluter. Constant false alarm rate detection of point targets using distribuited sensors. *Optical Engeneering, 37(2)*, pages 401–426, 1998.
50. A. Lazaro and J.R. Aranda. Multisensorial fusion for optimal object recognition. *Ultrasonics Symposium, 2000 IEEE Volume: 1*, pages 797 – 800, 2000.
51. S. Le Hegarth-mascle, I. Bloch, and D. Vidal-madjar. Introduction of neighborhood information in evidence theory and application to data fusion of radar and optical images with partial could cover. *Pattern Recognition, 31(11)*, pages 1811–1823, 1998.
52. S. Leader and R. Friend. A probabilistic, diagnostic and prognostic system for engine health and usage management. *Aerospace Conference Proceedings, 2000 IEEE Volume: 6*, pages 185 – 192, 2000.
53. E. Lefevre, O. Colot, P. Vannoorenberghe, and D. de Brucq. Knowledge modeling methods in the framework of evidence theory: an experimental comparison for melanoma detection. *Systems, Man, and Cybernetics, 2000 IEEE International Conference on Volume: 4*, pages 2806 – 2811, 2000.
54. E. Lefevre, O. Colot, P. Vannoorenberghe, and D. de Brucq. Use of fuzzy clustering for determining mass functions dempster-shafer theory. *Signal Processing Proceedings, 2000. WCCC-ICSP 2000. 5th International Conference on Volume: 3*, pages 1462 – 1470, 2000.

55. II Liggins, M.E., Chee-Yee Chong, I. Kadar, M.G. Alford, V. Vannicola, and S. Thomopoulos. Distributed fusion architectures and algorithms for target tracking. *Proceedings of the IEEE Volume: 85 Issue: 1*, pages 95 – 107, 1997.
56. Åsa Rönnbom Lisa Andersson. Intelligent agents - a new technology for future distributed sensor systems? Master's thesis, Department of Informatics, Goteborg University, 1999.
57. P. Maes. Agents that reduce work and information overload. *Communication of the ACM 38, 11*, pages 108 – 114, 1994.
58. Maja J Mataric. Behavior-based robotics. *MIT Encyclopedia of Cognitive Sciences Robert A. Wilson and Frank C. Keil, eds., MIT Press*, pages 74–77, 1999.
59. R.R. Murphy. Dempster-shafer theory for sensor fusion in autonomous. *Robotics and Automation, IEEE Transactions on Volume: 14 Issue: 2*, pages 197 – 206, 1998.
60. R.R. Murphy. Sensor and information fusion improved vision-based vehicle guidance. *IEEE Intelligent Systems [see also IEEE Expert] Volume: 13 Issue: 6*, pages 49 – 56, 1998.
61. H. Najjaran, N. Kircanski, and A.A. Goldenberg. Map building for a terrain scanning robot. *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on Volume: 4*, pages 3728 – 3733, 2001.
62. A. Nejatali and I. R. Ciric. Novel image fusion methodology using fuzzyset theory. *Optical engineer, 37(2)*, pages 485–491, 1998.
63. K.C. Ng and M.M. Trivedi. A neuro-fuzzy controller for mobile robot navigation and multirobot convoying. *Systems, Man and Cybernetics, Part B, IEEE Transactions on Volume: 28 Issue: 6*, pages 829 – 840, 1998.
64. Y.M. Niu, Y. S. Wong, and G. S. Hong. An intelligent sensorsystemapproachfor reliabletool flank wear recognition. *Int. Journal of Advanced Manufacturing Technology, 14(2)*, pages 77–84, 1998.
65. H. S. Nwana. Software agents: an overview. *Knowledge engineer review, 11, 3*, pages 1–40, 1996.
66. Lynne E. Parker. Alliance: An architecturefor fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation, 14(2)*, pages 220–240, 1998.
67. C. J. Petrie. Agent-based engineering, the web, and intelligence. *IEEE Expert/Intelligent Systems and Their Applications 11 6*, pages 24 – 29, 1996.
68. C Pohl and J. L. Van Genderen. Multisensor image fusion in remote sensing: concepts, methods and applications. *Int. Journal of Remote Sensing, 19(5)*, pages 823–854, 1998.
69. Hairong Qi, lyengar, S.S., and K. Chakrabarty. Distributed multi-resolution data integration using mobile agents. In *Aerospace Conference, 2001, IEEE Proceedings. , Volume: 3 , 2001*, pages 1133 –1141. IEEE, 2001.
70. Du Qingdong, Xu Lingyu, and Zhao Hai. D-s evidence theory applied to fault diagnosis of generator based on embedded sensors. *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on Volume: 1*, 2000.
71. Huber R. Information fusion for airborne insar observations.
72. C. Rago, P. Willett, and M. Alford. Predetection fusion: resolution cell grid effects. *Aerospace and Electronic Systems, IEEE Transactions on Volume: 35 Issue: 3*, pages 778 – 789, 1999.
73. V. Regis and W Thomas. Implementing soft real-time agent control. 2000.

74. M.J. Roemer, G.J. Kacprzynski, and M.H. Schoeller. Improved diagnostic and prognostic assessments using health management information fusion. *AUTOTESTCON Proceedings, 2001. IEEE Systems Readiness Technology Conference*, pages 365 – 377, 2001.
75. Liu Rujie and Yuan Baozong. A d-s based multi-channel information fusion method using classifier's uncertainty measurement. *Signal Processing Proceedings, 2000. WCCC-ICSP 2000. 5th International Conference on Volume: 2*, pages 1297 – 1300, 2000.
76. C. Scheering and A. Knoll. A framework for implementing self-organising task-oriented multi-sensor networks.
77. E. Shahbazian, R. Hallsworth, and D. Turgeon. Target tracking and identification issues when using real data. *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, pages 3 – 8, 2000.
78. B. Solaiman, R. Debon, F. Pipelier, J.-M. Cauvin, and C. Roux. Information fusion, application to data and model fusion for ultrasound image segmentation. *Biomedical Engineering, IEEE Transactions on Volume: 46 Issue: 10*, pages 1171 – 1175, 1999.
79. B. Solaiman, L.E. Pierce, and F.T. Ulaby. Multisensor data fusion using fuzzy concepts: application to land-cover classification using. *Geoscience and Remote Sensing, IEEE Transactions on Volume: 37 Issue: 3 Part: 1*, pages 1316 – 1326, 1999.
80. P. Stone and M. Veloso. Multi-agent systems: A survey from a machine learning perspective. *Technical Report 193,Department of Computer Science,CMU*, 1997.
81. A.W. Stroupe, M.C. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on Volume: 2*, pages 1092 – 1098, 2001.
82. T. Sundic, J. Marco, S.; Samitier, and P. Wide. Electronic tongue and electronic nose data fusion in classification with neural networks and fuzzy logic based models. *Instrumentation and Measurement Technology Conference, 2000. IMTC 2000. Proceedings of the 17th IEEE Volume: 3*, pages 1474 – 1479, 2000.
83. Xiaoou Tang. Multiple competitive learning network fusion for object classification. *Systems, Man and Cybernetics, Part B, IEEE Transactions on Volume: 28 Issue: 4*, pages 532 – 543, 1998.
84. R. Thrapp, C. Westbrook, and D. Subramanian. Robust localization algorithms for an autonomous campus tour guide. *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on Volume 2*, pages 2065 – 2071, 2001.
85. N.C. Tsourveloudis, K.P. Valavanis, and T. Hebert. Autonomous vehicle navigation utilizing electrostatic potential fields and fuzzy logic. *Robotics and Automation, IEEE Transactions on Volume: 17 Issue: 4*, pages 490 – 497, 2001.
86. F. Tupin, I. Bloch, and H. Maitre. A first step toward automatic interpretation of sar images using evidential fusion of several structure detectors. *Geoscience and Remote Sensing, IEEE Transactions on Volume: 37 Issue: 3 Part: 1*, pages 1327 – 1343, 1999.
87. L. Valet, G. Mauris, and P. Bolon. A statistical overview of recent literature in information fusion. *IEEE Aerospace and Electronics Systems Magazine Volume 16 issue 3*, pages 7 – 14, 2001.
88. L. Wald. Some terms of reference in data fusion. *Geoscience and Remote Sensing, IEEE Transactions on Volume: 37 Issue: 3 Part: 1*, pages 1190 – 1193, 1999.

89. Dayou Wang, J.M. Keller, C.A. Carson, K.K. McAdo-Edwards, and C.W. Bailey. Use of fuzzy-logic-inspired features to improve bacterial recognition through classifier fusion. *Systems, Man and Cybernetics, Part B, IEEE Transactions on Volume: 28 Issue: 4*, pages 583 – 591, 1998.
90. M. Wooldridge and P. Ciancarini. Agent oriented software engineering: The state of the art. 2000.
91. R. R. Yager. New modes of the owa information fusion. *Int Journal of Intelligent Systems 13(7)*, pages 661–681, 1998.
92. Zou Yi, Ho Yeong Khing, Chua Chin Seng, and Zhou Xiao Wei. Multi-ultrasonic sensor fusion for mobile robots. *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 387 – 391, 2000.
93. Zili Zhang and Chengqi Zhang. Result fusion in multi-agent systems based on owa operator. *Computer Science Conference, 2000. ACSC 2000. 23rd Australasian*, pages 234 – 240, 1999.